IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 09/915/417
Confirmation No. 7018
Filing Date 07/27/2001
Inventorship ....W. Allen
Assignee ....Hewlett-Packard Company
Group Art Unit 2624
Examiner ....Tommy D. Lee
Attorney's Docket No. 60980079-2
Title: Dynamic Generation of Linearized Halftone Matrix

DECLARATION UNDER 37 C.F.R. § 1.131

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name.

I believe I am an original, first, and joint inventor of the subject matter that is claimed and for which a patent is sought on the invention entitled "Dynamic Generation of Linearized Halftone Matrix," as identified above.

It is hereby asserted that the invention was conceived and actually reduced to practice in the United States or Spain prior to March 7th, 2000.

March 29th, 2000, is the filing date of the Ushiroda U.S. Patent (No. 6,831,756). Foreign application priority data includes a March 7th, 2000, Japan Patent Application (No. 2000-062271) and a March 30th, 1999, Japan Patent Application (No. 11-090068). The Japan Patent Application No. 11-090068 is subject to the pre-AIPA version of 35 U.S.C. 102(e). Therefore, the effective priority date under 102(e) is March 7th, 2000.

Accompanying this Declaration is Exhibits A through H. Exhibit A is a four-page redacted Invention Disclosure Form document. The "Outline of Invention" section of the Invention Disclosure Form in combination with Exhibits "B" through "H" evidences an actual reduction to practice. It is asserted that the redacted dates in the Exhibits are prior to March 7th, 2000.

All statements made herein of my own knowledge are true, and all statements made on information and belief are believed to be true. Further, these statements are made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. 1001) and may jeopardize the validity of the application or any patent issuing thereon.

* * * * * * * * * *

Full name of inventor:	William J. Allen
Inventor's Signature:	________________________
	Date: ____________________
Residence:	Corvallis, OR
Citizenship:	USA
Post Office Address:	3415 SW Cascade Avenue Corvallis, OR 97333-1533 USA

* * * * * * * * * *

* * * * * * * * * *

Full name of inventor:	Johan Lammens
Inventor's Signature:	Johan A. Lammens
	Date: January 12, 2006
Residence:	Sant Cugat del Valles
Citizenship:	Belgium
Post Office Address:	Hewlett-Packard ICD R&D Department Av. Graells, 501 08190 Sant Cugat (Barcelona) Spain

* * * * * * * * * *

HEWLETT PACKARD

INVENTION DISCLOSURE

Company Confidential: The information contained in this document is COMPANY CONFIDENTIAL and may not be disclosed to others without prior authorization.

PDNO 60980079 | Attorney PY

Date Received: [redacted]

Please submit this disclosure to the HP Legal Department as soon as possible. No patent protection is possible until a patent application is authorized, prepared and submitted to the Government.

Descriptive title of invention: DYNAMIC GENERATION OF A LINEARIZED HALFTONE MATRIX

Name of project: STORK/PIERO

Product name or number:

The invention is described in my lab book # ______ or in the following other records: NONE

Has the invention been built or tested? No [X] Yes [ ] Date built or tested: ______

Was a description of the invention published or are you planning to publish? No [X] Yes [ ]

Date and location of any actual or proposed publication of the invention: ______

Was a product including the invention shown, announced or sold? No [X] Yes [ ] Date and location of any actual or proposed announcement, offer to sell or sale of the product: ______

Name and date of anyone outside HP to whom invention will be or has been disclosed: NONE

If any will occur within three months, please call your IP attorney or the Legal Department at BCD 582 2428

Government Contract: Was this invention made under a government contract? No [X] Yes [ ]

Government agency and contract number: ______

Description of Invention: Please preserve all records of the invention and attach additional pages giving the following:

Each additional page should be signed and dated by the inventor(s) and witness(es)

a. Problems solved by the invention. SEE ATTACHED
b. Prior solutions and their disadvantages (if available attach copies of relevant technical articles or patents).
c. Description of the construction and operation of the invention (include appropriate schematic, block and timing diagrams; drawings; samples; graphs; flowcharts; computer listings; etc.).
d. Advantages of the invention over what has been done before.

Signature of Inventor(s): PLEASE SEE BACK OF FORM FOR ADDITIONAL REQUIRED INVENTOR INFORMATION.

I(we) this date [redacted] submit this invention disclosure pursuant to my(our) employment agreement.

Emp. Number	Name	Signature	Telnet #	Mail Stop	Entity Name and Lab Name
241273	WILLIAM J. ALLEN	William J. Allen	712-2164		BCD R&D
402316	JOHAN LAMMENS	Johan M. Lammens	712-2469		BCD R&D

If more than three inventors, please sign on an additional disclosure form and attach to this page.

Please try to obtain the signature of the person(s) to whom the invention was first disclosed.

Signature of Witness(es):

The invention was first explained to and understood by me (us): [redacted]

Full Name	Signature	Date of signature
ANTONIO LAIN		[redacted]

Dynamic Generation of a Linearized Halftone Matrix

Invention Disclosure

Will Allen, HP BCD
Johan Lammens, HP BCD
[redacted]

Problem

It is desirable, from performance and image quality standpoints, to have a halftone matrix that is linear. Linear means the system response (e.g., CIE L^*) is linear with respect to digital counts input to the halftoning module. Since system response varies (environment, print cartridge set, medium, et cetera), it is desirable to have the ability to very the linearization depending on the state of the system.

A classic technique involves using a look up vector to change the input data before halftoning. The vector is changed depending on the linearization required. This technique often introduces unwanted contouring in the output plot. If the correction is strong, unique states of the input data are lost. For example, applying a gamma correction of 2.0 to 8-bit data causes ¼ of the 256 unique states to be lost!

Another approach is to generate a halftone matrix that is inherently linearized with respect to a given set of conditions. Since conditions vary, the matrix must be generated from time to time after the product leaves the factory. Some forms of halftone matrices are expensive, in terms of time and/or memory, to generate, thereby rendering them ill-suited to dynamic (in the product) generation.

Using a pre-defined matrix is not reasonable, as it would be prohibitive to store pre-defined matrices for all possible linearizations in the printer.

The invention details a scheme applicable to any type of traditional threshold matrix based halftoning that will rapidly generate a matrix with a given response (linearization) in terms of number of dots as a function of input in digital counts.

Outline of Invention

The idea is to obtain (generate, leverage, buy) a 16-bit traditional threshold matrix and store it in the printer in a special modified format. Let's call this special format a *level vector*. When needed, a traditional 8-bit threshold matrix is rapidly generated, with any given linearization, from the level vector data and a second vector specifying the desired linearization.

Translation to Level Vector Form

Although applicable to other size matrices, the example given here will be a 256 x 256 element matrix.

The first step is to obtain a traditional 256 x 256 x 16-bit threshold matrix and convert it into a level vector. Each element in the matrix is a 16-bit word, and can encode one of 2^{16} (65,536) different threshold values. This can be done with any means desired, and the style (Bayer, screen, et cetera) of the matrix is not important. The matrix has 2^{16} elements, and can therefore represent $2^{16} + 1$ unique numbers of dots. Each number of dots can be thought of as a *level*.

In a traditional threshold matrix, the positions of data in the matrix represent a positions on the page, while the values represent thresholds or levels. The new form stores the same information as a vector. The index, or position, in the vector indicates the threshold value, while the data stored under each index indicate the x and y coordinates, in the traditional threshold matrix, of the pixel "turned on" at the given threshold.

Will Allen [redacted]
William Allen

Johan R. Lammens [redacted]
Amparo Lain [redacted]

Here is a traditional threshold matrix. For simplicity a 4 x 4 matrix will be shown and translated into a level vector. The process is easily generalized to a matrix with larger dimensions. The numbers inside the cells are thresholds. The numbers outside the cells are the indices, or *x* and *y* coordinates, of the cells.

3	4	8	14	7
2	13	0	3	11
1	10	2	**1**	15
0	6	12	9	5
	0	1	2	3

Here are the same data represented as a level vector. The *x* data and *y* data columns are simply the coordinates of the threshold value in the traditional table above that has the same value as the Index of that coordinate pair in the level vector. The **bold** box surrounds the "same" data in both forms.

Index	*x* data	*y* data
0	1	2
1	**2**	**1**
2	1	1
3	2	2
4	0	3
5	3	0
6	0	0
7	3	3
8	1	3
9	2	0
10	0	1
11	3	2
12	1	0
13	0	2
14	2	3
15	3	1

The 4 x 4 traditional matrix is comprised of 16 8-bit values. It occupies 128 bits of memory. The level vector consists of 32 4-bit values and also occupies 128 bits of memory. No extra memory is required for the level vector.

Going back to the 256 x 256 x 16-bit case, we see the traditional threshold matrix has 2^{16} 16-bit entries and therefore occupies 2^{17} bytes of memory. The same data, in a level vector, require 2^{16} entries each composed of 2 8-bit values, or 2^{17} bytes.

Dynamic Generation of 8-bit Traditional Matrix

A general form of the linearization specification would be a vector of 256 16-bit elements. Each element defines the number of dots to be "turned-on" at the halftone level (input digital counts) corresponding to the index. For example, if the entry at index 13 was 873, then 873 dots in the 8-bit threshold matrix would be at or below the threshold value of 13.

To generate the 256 x 256 x 8-bit linearized matrix, one needs only to sequence through the linearization vector. If the 1st element of the linearization vector contains 12, then the first 12 locations specified by the level vector are assigned threshold value 1. For the 2nd element in the linearization vector, we'll assume the value is 47. The next 35 (47 – 12) locations specified by the level vector are assigned threshold value 2. This is repeated until all the values in the linearization vector have been processed, thereby assigning a threshold value to all the cells in the 256 x 256 x 8-bit matrix.

[illegible] [redacted]
[illegible] [redacted]

[redacted]

Advantages

- The level vector is easily computed by translation from a traditional 16-bit threshold matrix. It occupies the same amount of memory as the traditional matrix.
- The technique is applicable to any style of halftoning that can be represented in a traditional threshold matrix, including matrices purchased from third parties.
- Image quality is improved by eliminating contouring caused by vector based linearization applied before halftoning.
- Execution time and halftoning-time complexity are reduced when compared to vector based linearization applied before halftoning.
- Generation of the 8-bit threshold matrix is fast, and occurs in a deterministic amount of time that is almost completely independent of the data stored in the level and linearization vectors.
- A single level vector can be used to generate all 8-bit threshold matrices needed by the printer. This reduces storage requirements.
- The scheme is compatible with dynamic compensations systems such as Canguro's *Closed Loop Color.*

[illegible] [redacted]
[illegible] [redacted]

[redacted]

HT Color Tuning

Johan M. Lammens

[redacted]

Calculate four 1D transfer funcitons (C,M,Y,K) to minimize the color difference between two halftones, using the same ink, media, printmode, and printer, without affecting any other part of the system (color maps etc).

1. Using 1D transfer functions calculated from 4D profiles

Based on full colorimetric characterization (ICC profiles) and numerical minimization of color differences.

[redacted]

Create a 9^3 RGB sampling and save as raw image, to be transformed to CIELAB by PhP or so using AC intent; this will be the forward table used to characterize a given HT. This roundabout way of creating an AC forward table is for lack of a tool to extract it directly fromt an ICC profile. Note PhP uses inverted coding for CMYK (255=0%).

[redacted]

Read the corresponding Lab values for reference and new HT

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Construct a 3D interpolation (RGB->Lab map) using Mathematica's standard 3rd order interpolation, for both the reference HT and the new HT. It should be possible to use the measured data directly to do this.

```
SetOptions[ListInterpolation, InterpolationOrder → 1];
```

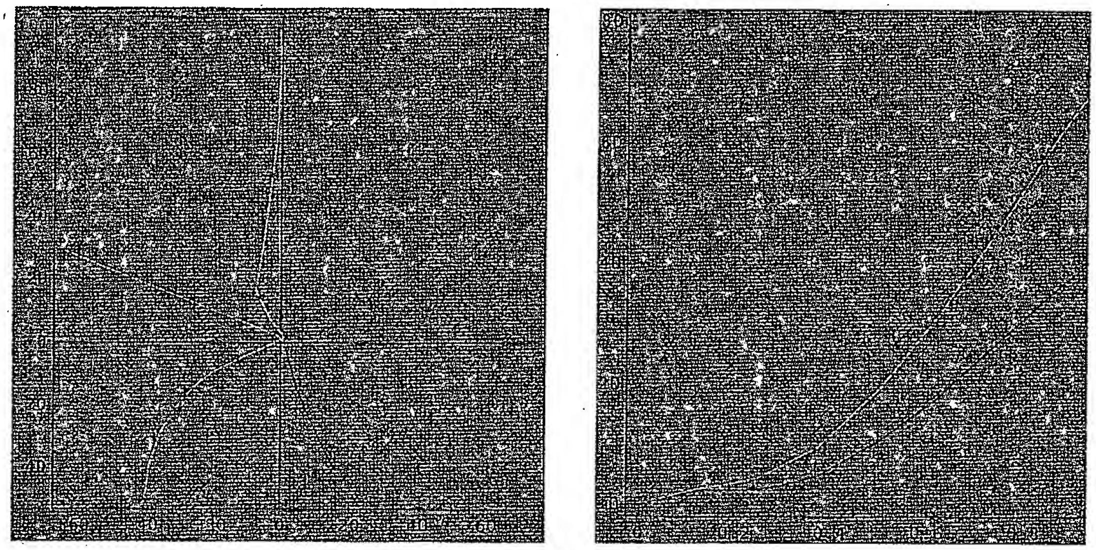
[redacted]

[redacted]

Plot primary and secondary ramps in ab space and dE from paper for primary ramps, for reference and new HT.

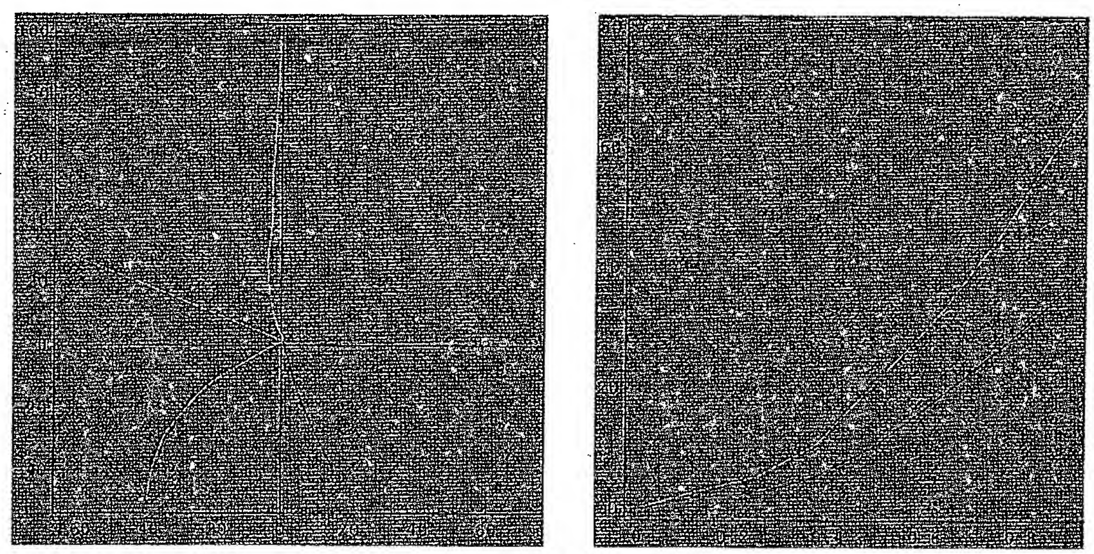
[redacted]

```
plref = PlotRampsRGB[refmod]
```



- GraphicsArray -

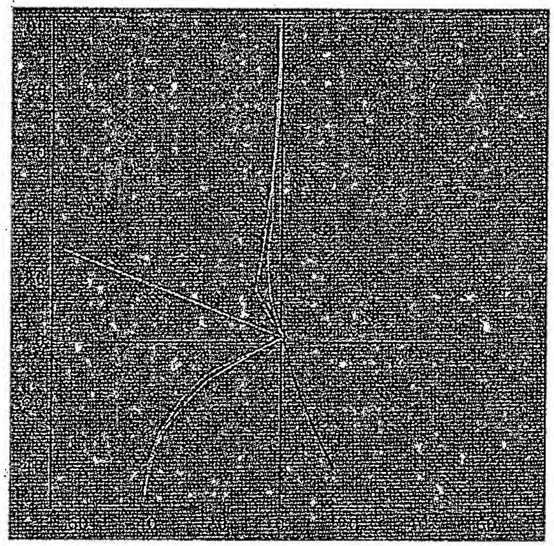
```
plnew = PlotRampsRGB[newmod]
```

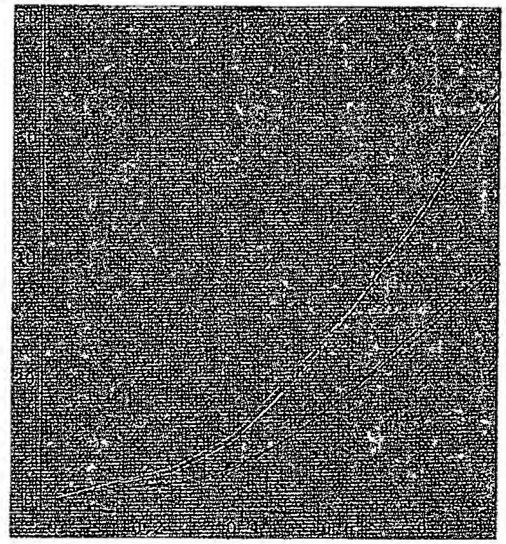


- GraphicsArray -

Now show the two together, to appreciate the difference better.

[redacted]



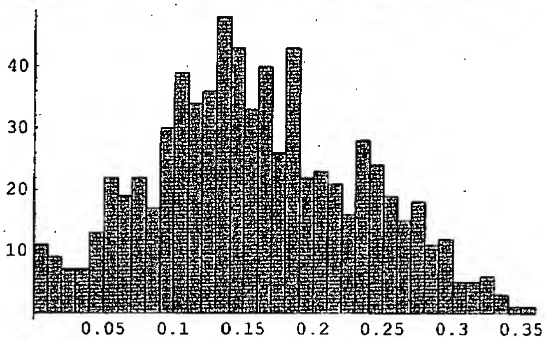


- GraphicsArray -

Compare model calculated values to the original IT8 measurements to get an idea of the accuracy of the interpolated model.

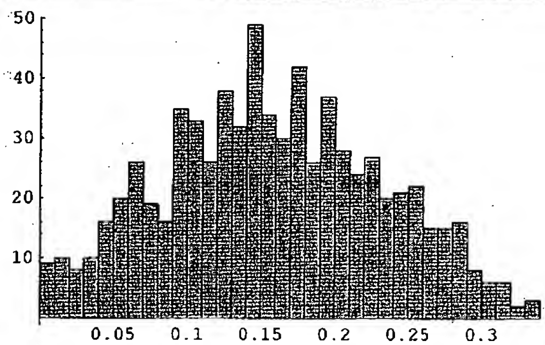
[redacted]

[redacted]



{0., 0.158767, 0.351407, 0.152195, 0.0743582}

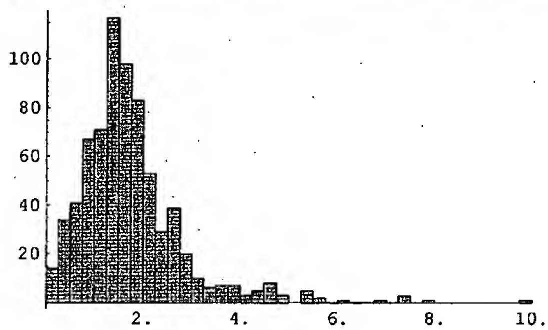
[redacted]



{0., 0.158309, 0.337512, 0.154722, 0.0737709}

Difference between old and new HT color data

[redacted]



{0.126667, 1.75705, 9.99575, 1.55574, 1.11485}

Gray balance the new HT while matching the color of the reference HT, using only the CMY planes. Using all four planes does not work well; the numerical minimization becomes unstable because the CMYK->Lab mapping is not unique (1-to-1). Mind the accuracy and precision goals - since we're working in dE space it makes no sense to use the default 10 bits or so; this would only cause the minimization to thrash about needlessly. Use only one of the three alternatives below: optimzing r,g,b separately; only r,g assuming b=r; or only r, assuming g=b=r.

Optimizing r,g,b separately

[redacted]

Optimizing r,g and assuming b=r

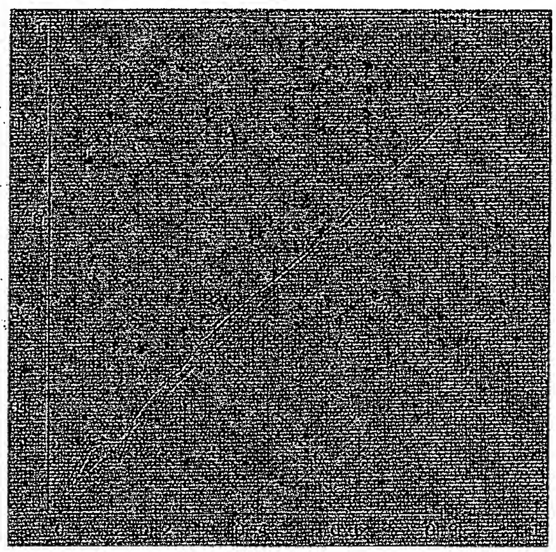
[redacted]

Optimizing r and assuming g=b=r

[redacted]

[redacted]

[redacted]



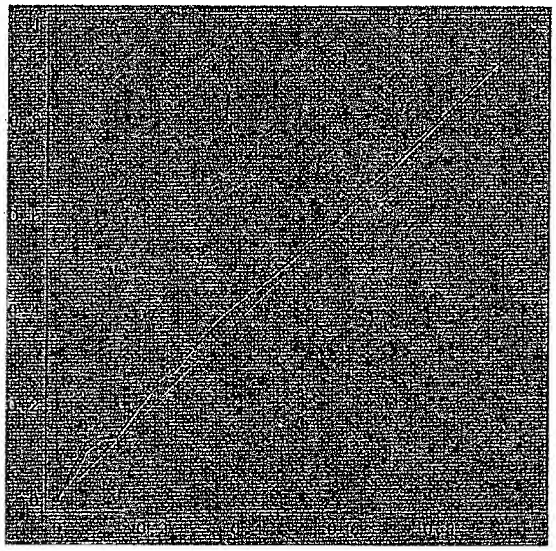
- Graphics -

Force the result to be monotonically increasing!

[redacted]

[redacted]

[redacted]



- Graphics -

Write the resulting tables out in Mathematica, ABS ScreenMaker and/or dot histogram format

[redacted]

[redacted]

[redacted]

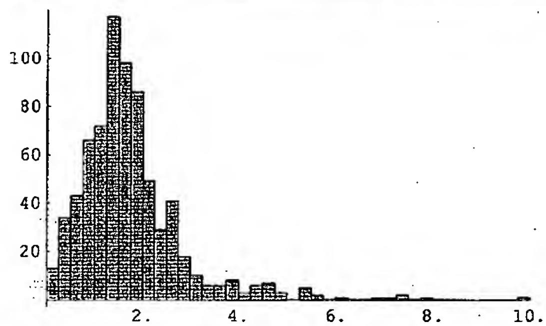
[redacted]

Caculate color differences over an 11^3 RGB target before and after linearization (color correction), but without reprinting and remeasuring yet, and using the models rather than the measurement data.

[redacted]

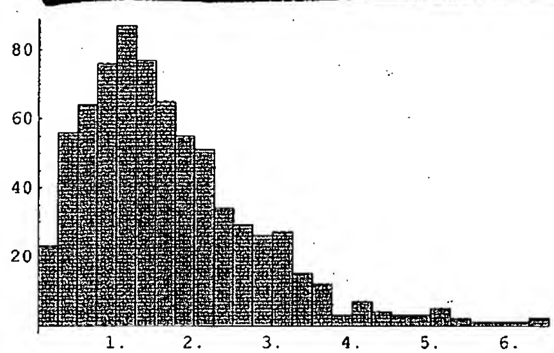
[redacted]

[redacted]



{-Graphics-, 1.75224, 9.94129, 1.55618, 1.10944}

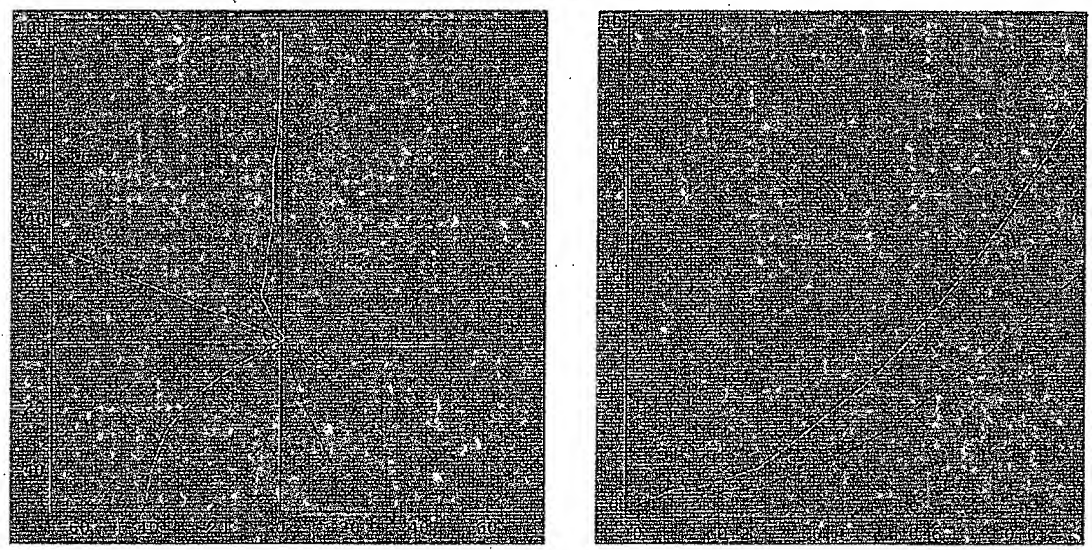
[redacted]



{-Graphics-, 1.67048, 6.39818, 1.46681, 1.09276}

Plot primary and secondary ramps as before, but after applying the linearization functions.

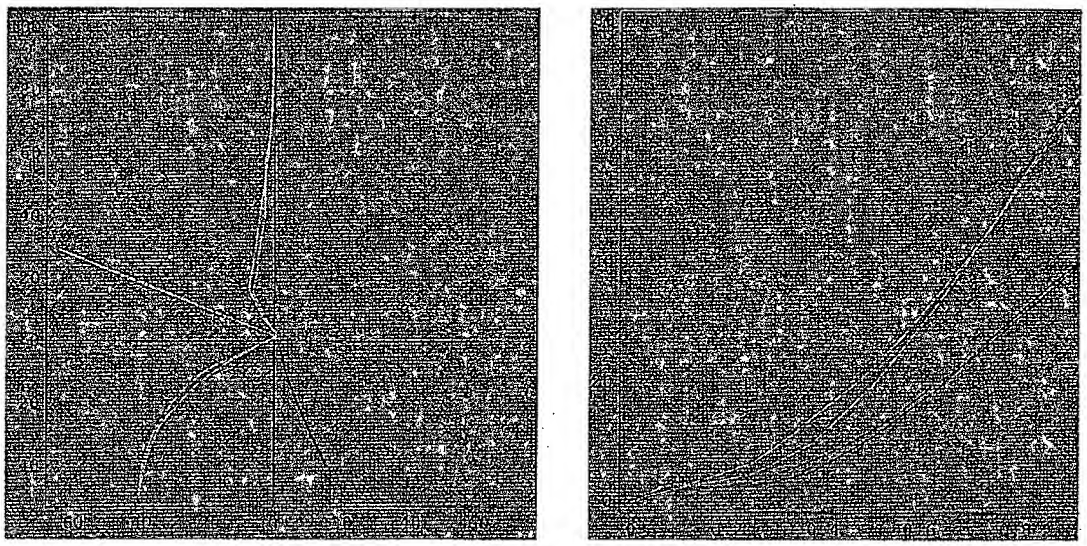
[redacted]



- GraphicsArray -

Show reference and result together. Note that there may be small difference induced in the indidvidual ramps in order to minimize the differences in the gray balance. Simple 1D linearization of the primaries alone would probably make the primary ramps match better, but potentially at the cost of gray balance and overall color matching.

[redacted]



- GraphicsArray -

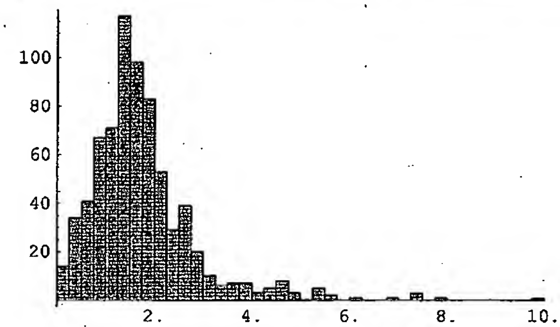
At this point we could write out a new set of "measurement" data to create a new profile corresponding to the linearized/color corrected HT. It doesn't have to be in IT8 form; since we can just turn the model crank we can generate an arbitrary number of samples, e.g. 11^4 (which would be rather time consuming to actually measure). That is, if you have faith in the model ;-)

The tranfer functions could be put into the new profile as output tables, but it's better to build them right into the (matrix) HT to avoid losing gray levels, especially if a higher bit depth (10 or 12b) version is avaiable to convert down to 8 bits while massaging the histogram to fit the required linearization. For non-matrix HTs this is more tricky.

Finally, after applying the linearizations to the new HT and reprinting the IT8 target with it, read in the measured Lab values and compare to the reference ones. This is the ultimate test for how well the model works, of course. If all is well, the results should be comparable to the ones above, which were obtained from the models only (without measurements).

[redacted]

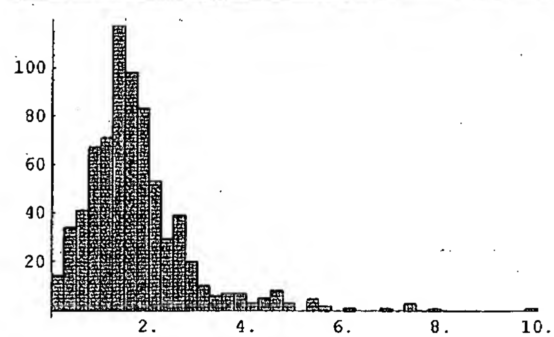
[redacted]



{0.126667, 1.75705, 9.99575, 1.55574, 1.11485}

Select the parts of the IT8 target that are below the global ink limit for the device/ink/media under consideration, to get a more realistic picture of the color differences one might see in practice.

[redacted]



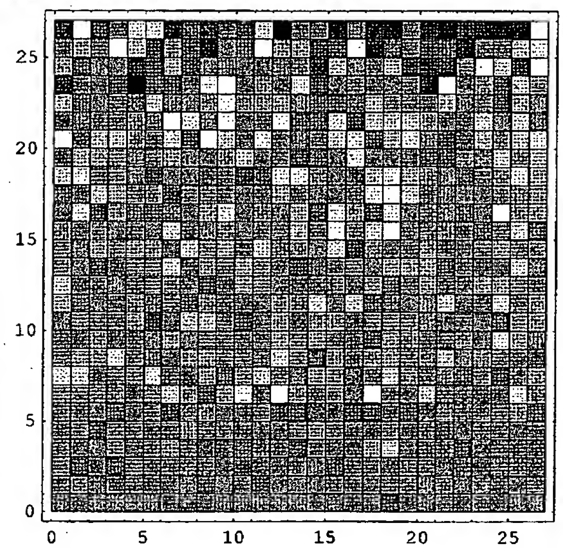
{0.126667, 1.75705, 9.99575, 1.55574, 1.11485}

Sort the samples by descending dE to get an idea of the worst case colors

[redacted]

The following plot shows the samples with the least color differences at the top, the worst at the bottom, each colored with the corresponding CMYK color.

[redacted]



- DensityGraphics -

2. Using 1D transfer functions calculated from 1D ramps

Now compare what the result would be using traditional 1D linearizations, using the model data only (no measurements).

define the input dot percentages for a 9x21 characterization target

[redacted]

generate CIELAB data for reference and new HT, using their respective models, and convert to dE from paper

[redacted]

[redacted]

[redacted]

[redacted]

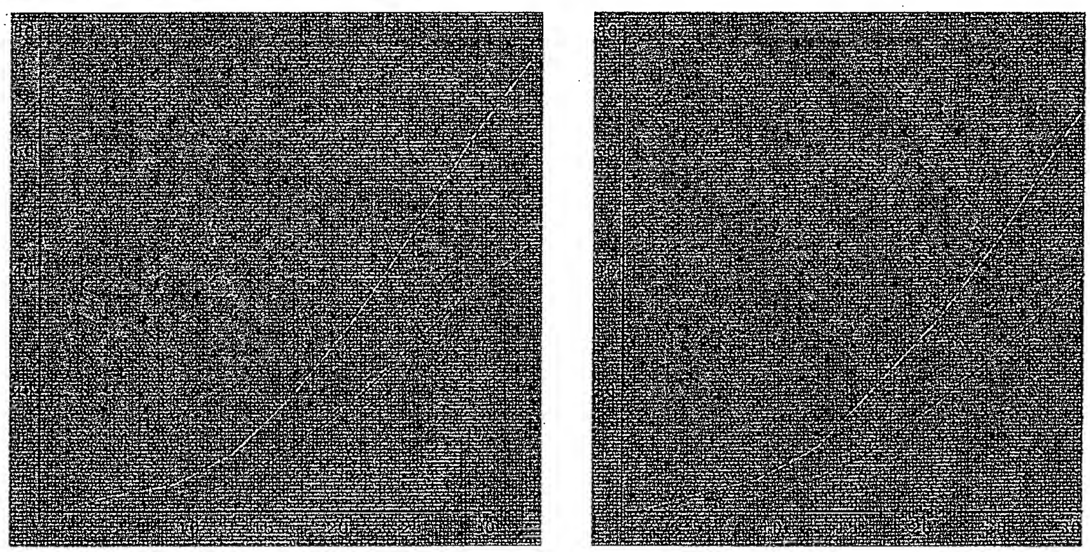
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



```
- GraphicsArray -
```

Convert to {x,y} format data using the ramp values

[redacted]

Now calculate linearization functions, using a quick and dirty numerical function inverse of interpolated functions. Varying the interpolation order will determine the smoothness of the resulting transfer functions.

[redacted]

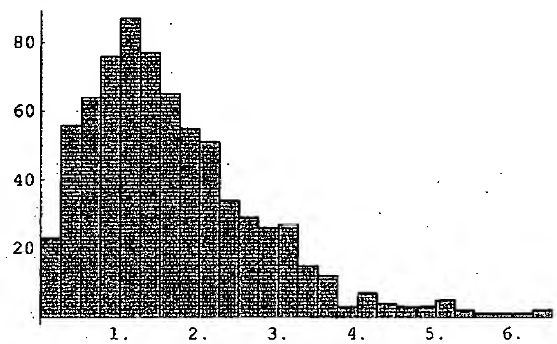
[redacted]

Caculate color differences over the IT8 target before and after linearization (color correction), but without reprinting and remeasuring yet, and using the models rather than the measurement data.

[redacted]

Previous result with gray balancing:

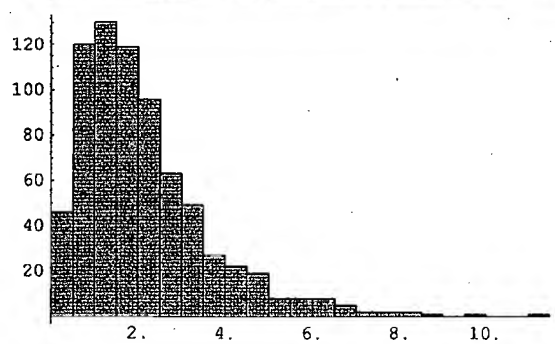
[redacted]



{- Graphics -, 1.67048, 6.39818, 1.46681, 1.09276}

New result with 1D linearizations:

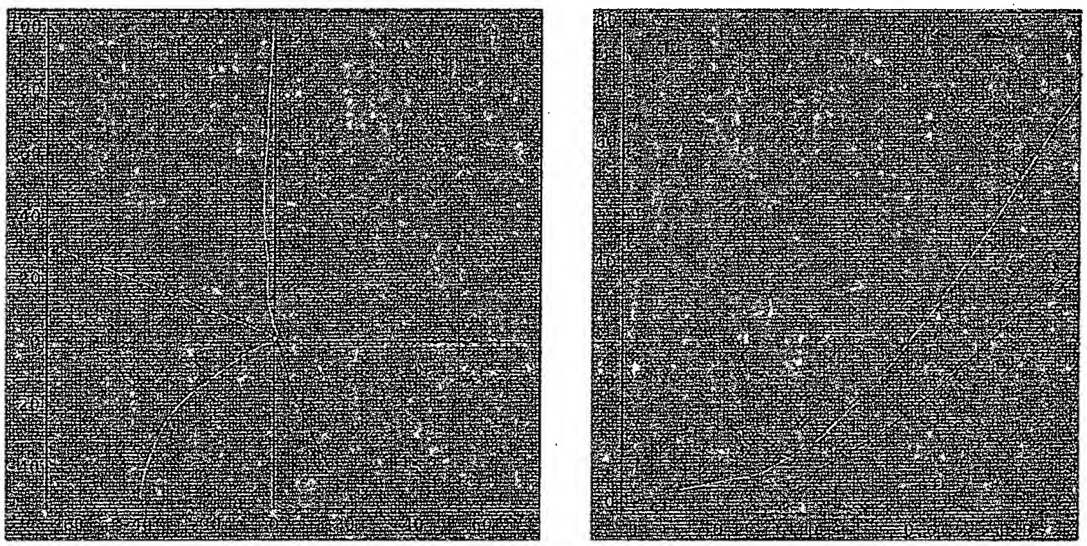
[redacted]



{- Graphics -, 2.12956, 11.1837, 1.75619, 1.51653}

Plot primary and secondary ramps as before, but after applying the new linearization functions.

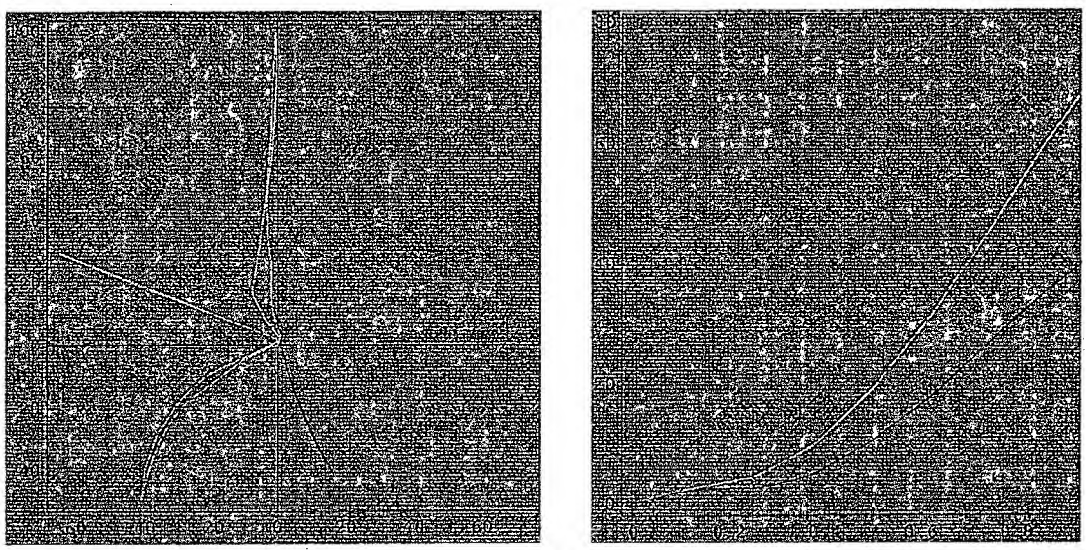
[redacted]



- GraphicsArray -

Show reference and result together. The primary ramps should match better than before.

[redacted]



- GraphicsArray -

HT dot distance

Johan M. Lammens

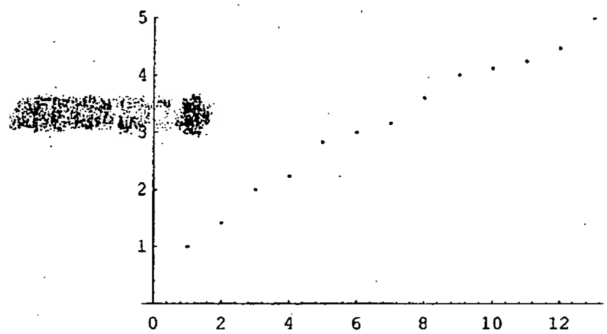
[redacted]

Read calculated dot distance profile from file; this one is for Canguro's default m256_lin BN halftone (calculated off-line with /users/lammens/Canguro/Correctdither/distprofile.c). Return value is a list of measured distances (in pixels), the data matrix, and interpolated functions per plane for number of pixels, cumulative pixel area, and average number of pixels at each distance, as a function of normalized digital count in [0,1].

[redacted]

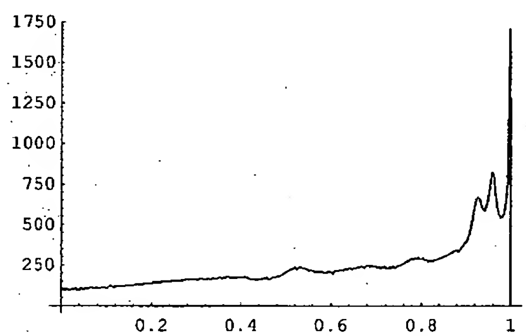
The first part of the result is the vector of measured distances: discrete and step-wise because of the regular grid of the the HT, and measured in pixels.

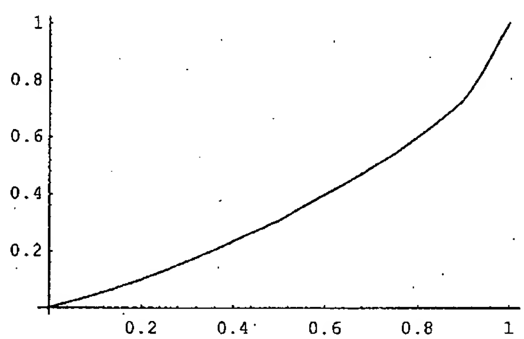
```
ListPlot[dist,PlotRange->{0,Max[dist]}];
```



This plots the number of pixels and cumulative pixel area as a function of normalized digital count, for the first plane of the HT

```
Plot[#[c],{c,0,1},PlotRange->All]& /@ Take[fns[[1]], 2];
```

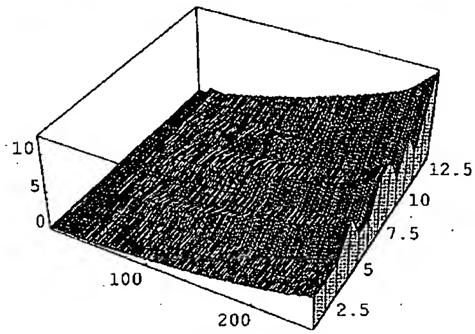


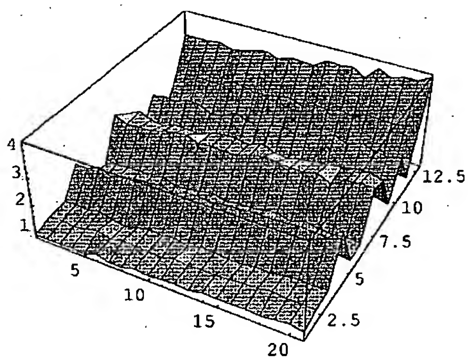


The cumulative pixel area is effectively the dot histogram of the halftone. Note that Alp halftones are in KCMY plane order. This HT is pre-linearized. Note also that the number of pixels added for each consecutive level becomes quite irregular near the shadow end, although the cumulative area goes up smoothly and monotonically.

The following plots show the number of pixels at each measured distance, as a function of normalized digital count, for the first plane, and a close-up around 50% pixel area. The effect of BN is visible as fluctuations around the monotonically increasing trend.

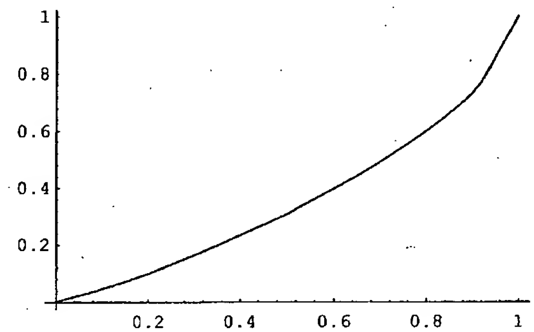
[redacted]

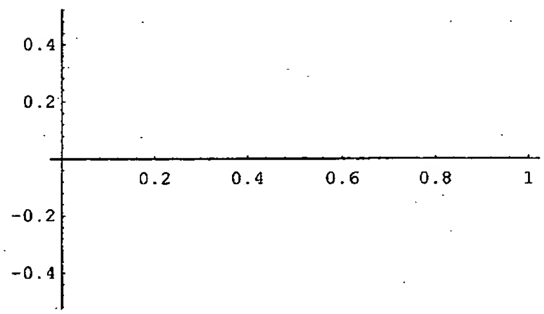




Function to return total simple and overlapped area coverage based on dot profile, normalized input digital count (nominal area coverage), and normalized dot size (expressed in square pixel diagonals).

[redacted]





[redacted]

HT pre-linearization

Johan M. Lammens

[redacted]

Read an N-bit HT matrix (N>8) and a desired dot profile (linearization function), and generate an 8 bit version of the HT with that dot profile.

Read the HT matrix: single plane, 16 bit little endian encoding

[redacted]

library files location

[redacted]

Read input HT: 8, 16, or 32 bit encoding

[redacted]

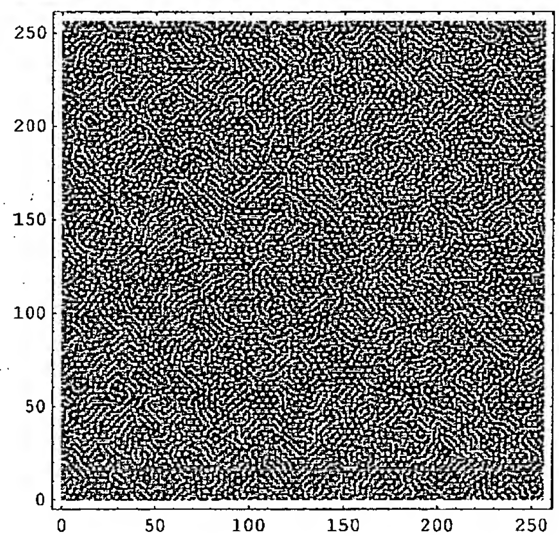
[redacted]

Optional: invert the thresholds in case there is doubt about additive vs subtractive interpretation

[redacted]

Display in 2D matrix form

[redacted]



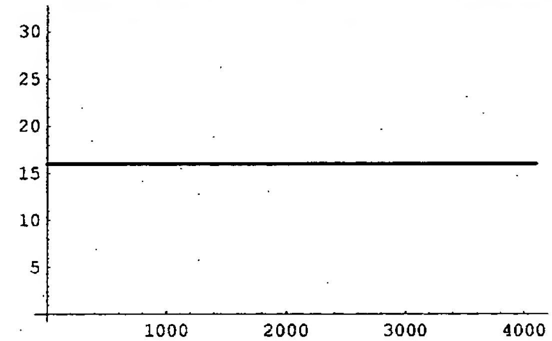
Write out in Alp monochrome form and for FFT analysis

[redacted]

[redacted]

build input histogram

[redacted]




```
{3.11667 Second, {{4096}, 65536}}
```

shortcut to histogram for dot-linear HTs

[redacted]

normalize by total dot area and replot

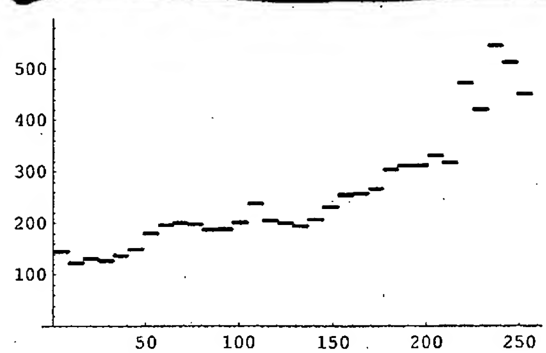
[redacted]

Read calculated dot distance profile and histogram from file; this one is for Canguro's default m256_lin BN halftone (calculated off-line with /users/lammens/Canguro/Correctdither/distprofile.c). Return value is a list of measured distances (in pixels), the data matrix, and interpolated functions per plane for number of pixels, cumulative pixel area, and average number of pixels at each distance, as a function of normalized digital count in [0,1].

[redacted]

select the appropriate plane's dot histogram and plot it - note that the order is KCMY (for Alp)

[redacted]



{{256}, 65536}

normalize by total dot area and replot

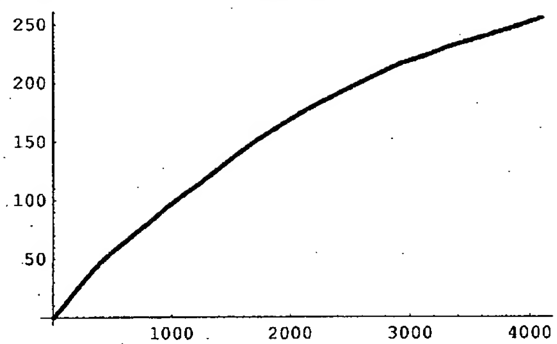
[redacted]

Write out the normalized dot (threshold) histogram and cumulative histogram as real vectors

[redacted]

Build a mapping from 16 to 8 bit threshold values, based on the two histograms.

[redacted]



0

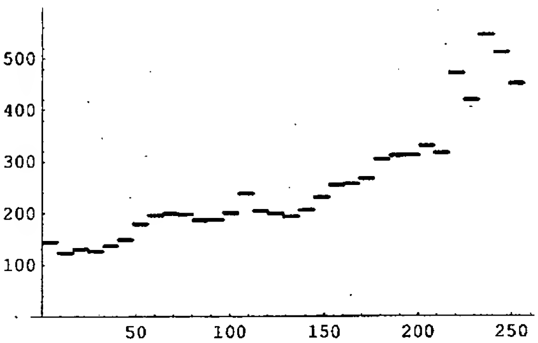
test some values along the tone curve, specified in % ink

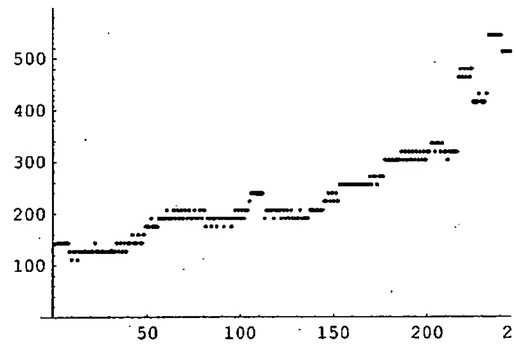
[redacted]

create the new 8 bit HT and check the dot histogram and grayscale representation

[redacted]

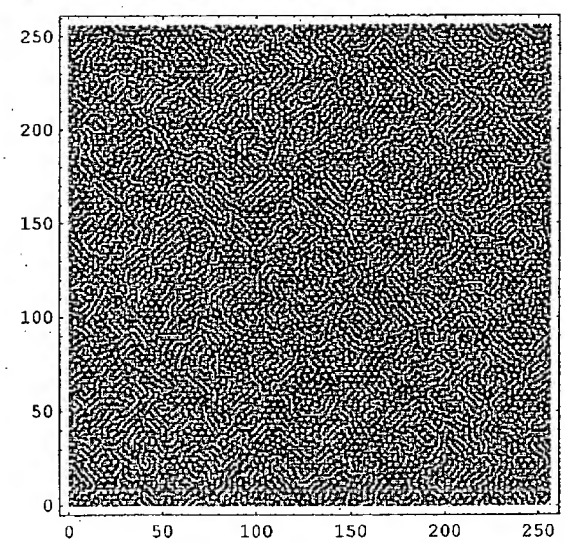
[redacted]

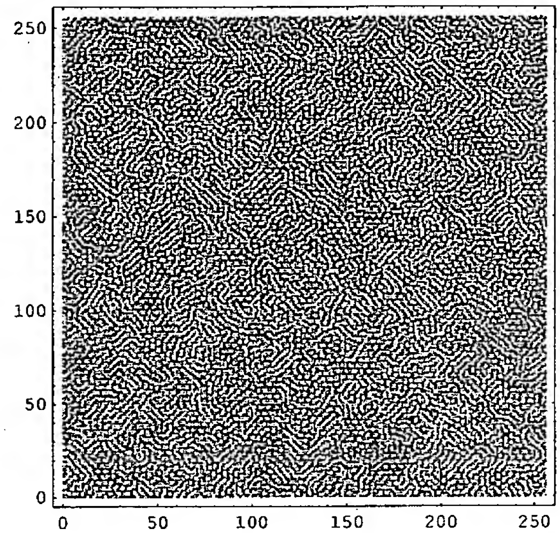





```
{{256}, 65536}
```

[redacted]





Write out the new HT to file

[redacted]

[redacted]

Convert raw format threshold matrices to Alp and other formats.

Johan M. Lammens [redacted]

Raw to Alp monochrome format

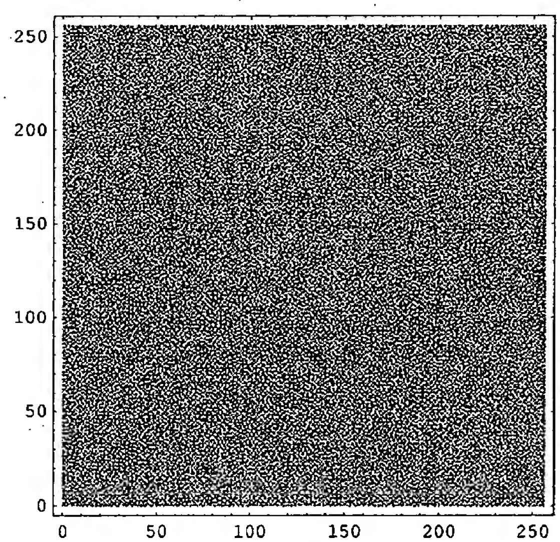
[redacted]

Read input HT, 8 or 16 bit

[redacted]

Display in 2D matrix form

[redacted]



Write out in Alp monochrome form and for FFT analysis

[redacted]

Raw to Alp four plane format. Note that Alp HTs are inverted (RGB interpretation).

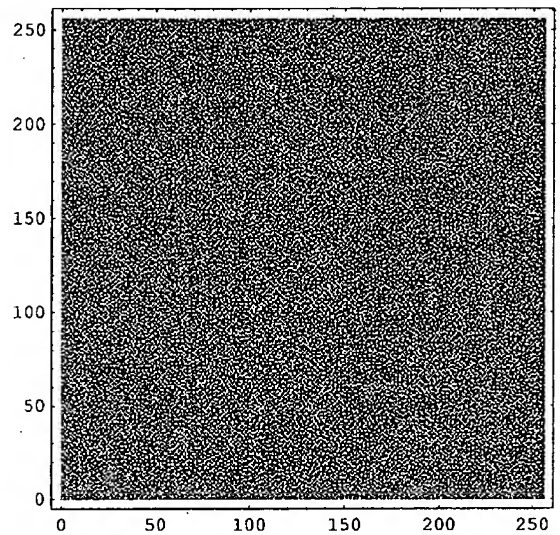
[redacted]

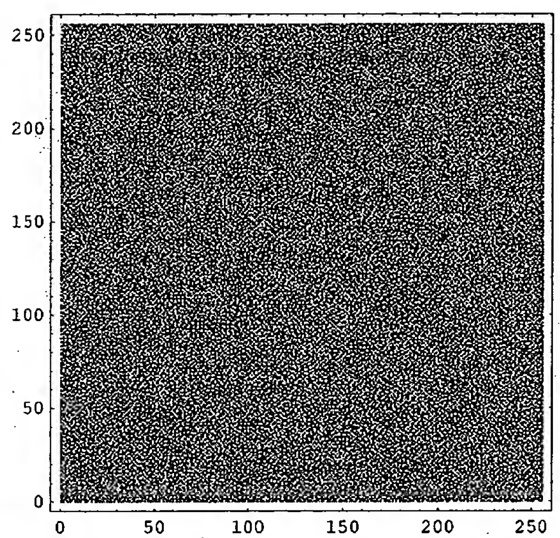
Read input HT, 8 or 16 bit

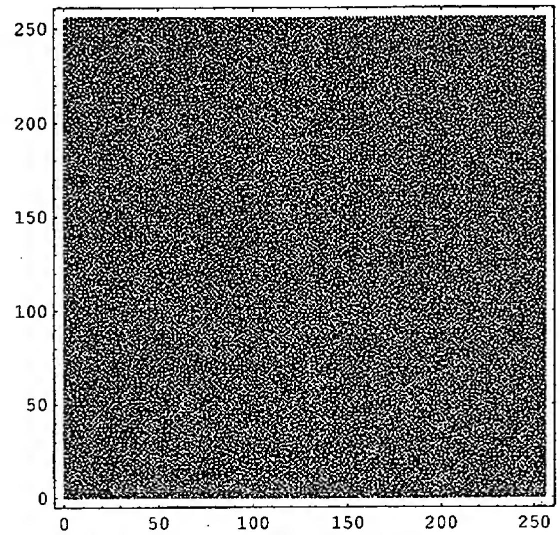
[redacted]

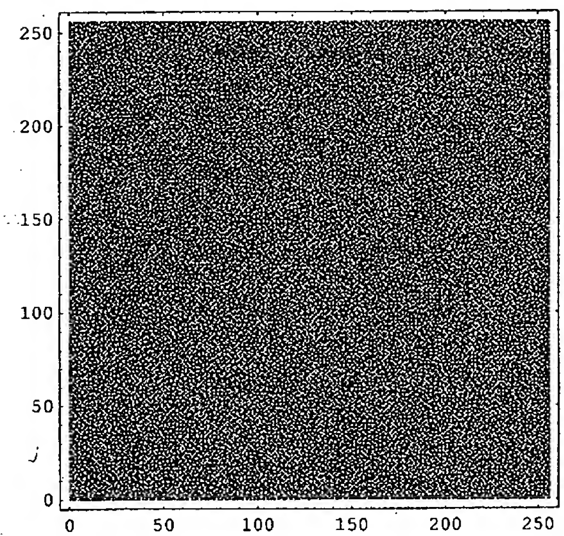
Display in 2D matrix form

[redacted]


250
200
150
100
50
0
0
50
100
150
200
250


250
200
150
100
50
0
0
50
100
150
200
250






```
{- DensityGraphics -, - DensityGraphics -, - DensityGraphics -, - DensityGraphics -}
```

Write out in 4 plane Alp format, for use with Canguro Alp etc. NOTE: KCMY format! Planes are rotated inside the function.

[redacted]

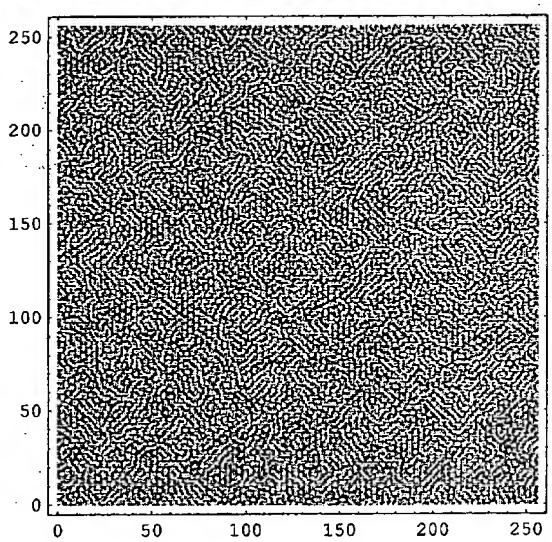
Alp format to raw

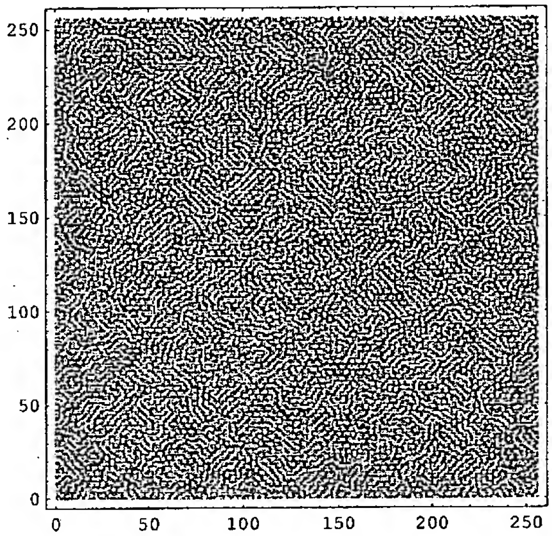
[redacted]

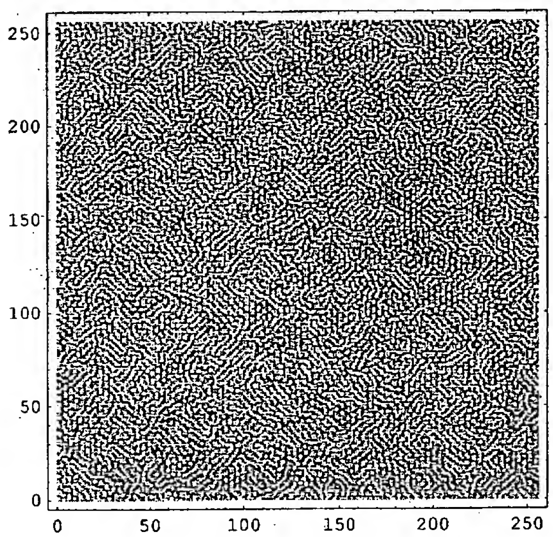
[redacted]

Raw to C constant-like text format (really mma expression format)

[redacted]







Note HPGL plane order is KCMY; rotate the planes before writing if 4-plane matrix!

[redacted]

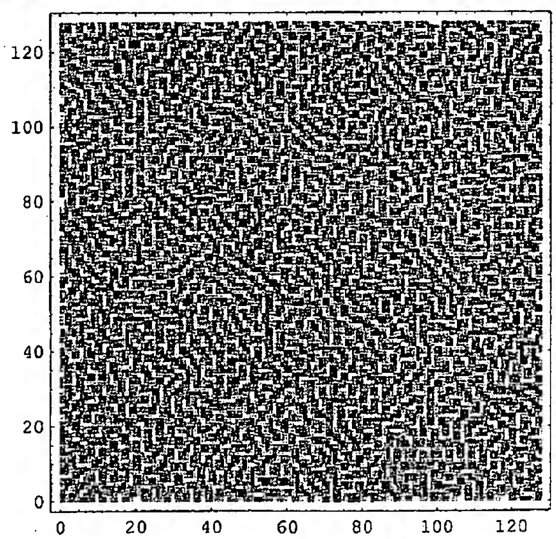
[redacted]

Raw >8b to 8b raw, linear scaling

[redacted]

Optional: invert polarity

[redacted]



[redacted]

HT Color Tuning

Johan M. Lammens

[redacted]

Calculate four 1D transfer funcitons (C,M,Y,K) to minimize the color difference between two halftones, using the same ink, media, printmode, and printer, without affecting any other part of the system (color maps etc).

1. Using 1D transfer functions calculated from 4D profiles

Based on full colorimetric characterization (ICC profiles) and numerical minimization of color differences.

[redacted]

Create an 11^4 CMYK sampling and save as raw image, to be transformed to CIELAB by PhP or so using AC intent; this will be the forward table used to characterize a given HT. This roundabout way of creating an AC forward table is for lack of a tool to extract it directly fromt an ICC profile. Note PhP uses inverted coding for CMYK (255=0%).

[redacted]

Read the same image back, converted to Lab using and external CMM and ICC profile corresponding to the reference and new HT using the same device configuration. Convert from PhP raw to Lab coding. PhotoShop Lab range is [0,100], [-128,127], [-128,127], mapped to 8 bit values as [0,255], with 0->128 for a and b.

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Construct a 4D interpolation (CMYK->Lab map) using Mathematica's standard 3rd order interpolation, for both the reference HT and the new HT. It should be possible to use the measured data directly to do this, but I haven't figured out how to deal with the non-uniform sampling of the IT8 target yet.

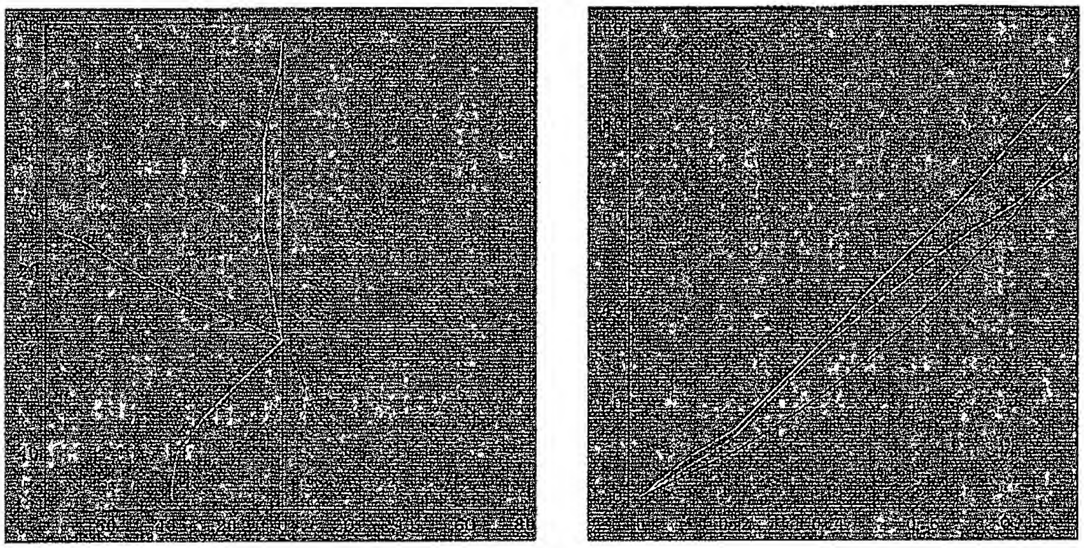
[redacted]

[redacted]

Plot primary and secondary ramps in ab space and dE from paper for primary ramps, for reference and new HT.

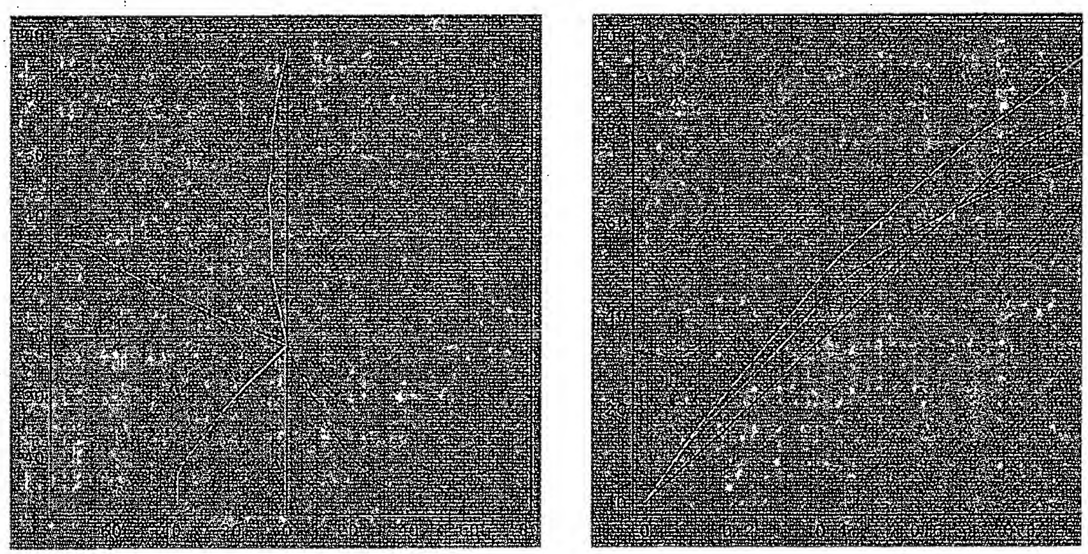
[redacted]

[redacted]



- GraphicsArray -

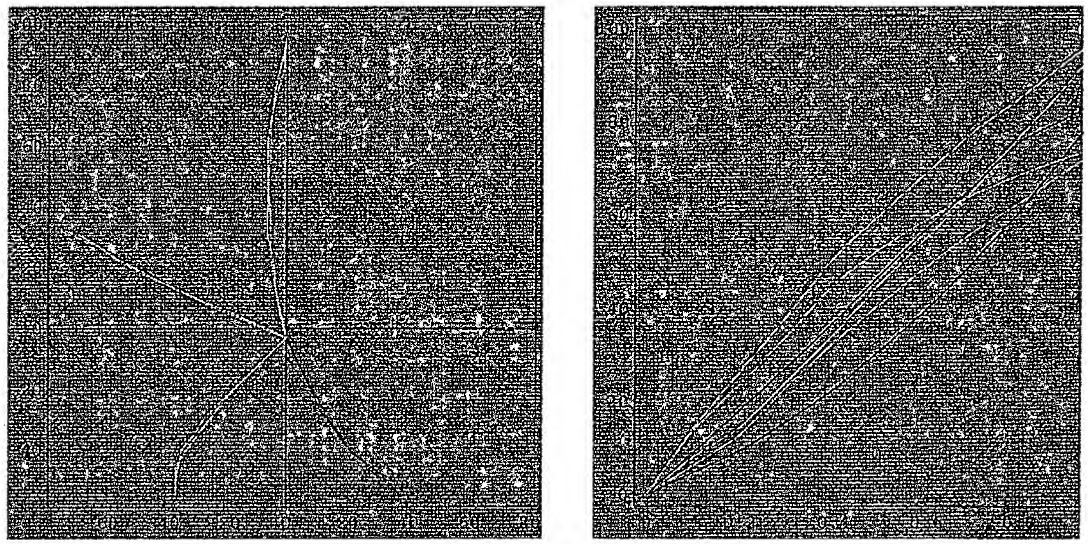
[redacted]



- GraphicsArray -

Now show the two together, to appreciate the difference better.

[redacted]

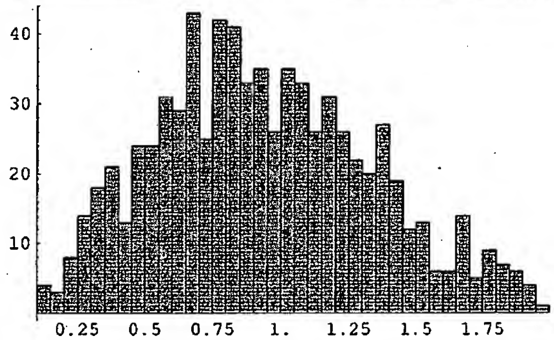


- GraphicsArray -

Compare model calculated values to the original IT8 measurements to get an idea of the accuracy of the interpolated model.

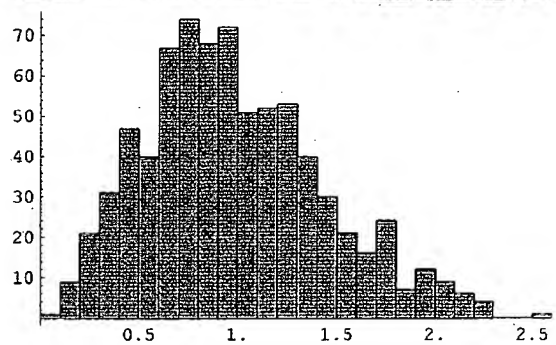
[redacted]

[redacted]



{0.125059, 0.942216, 1.9738, 0.909561, 0.399198}

[redacted]



{0.065127, 0.983044, 2.50497, 0.920793, 0.450623}

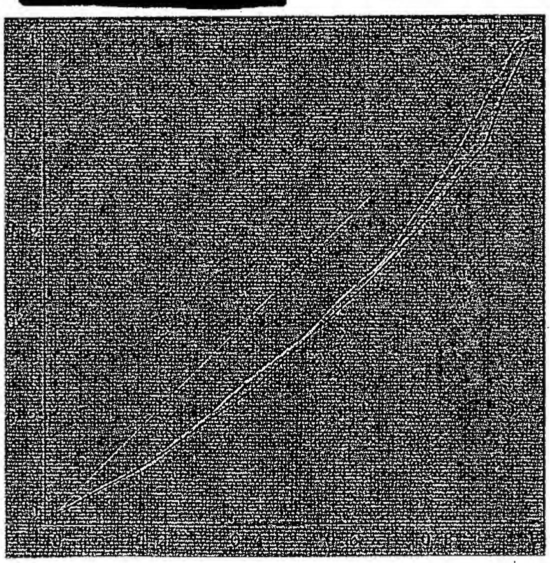
Gray balance the new HT while matching the color of the reference HT, using only the CMY planes. Using all four planes does not work well; the numerical minimization becomes unstable because the CMYK->Lab mapping is not unique (1-to-1). Mind the accuracy and precision goals - since we're working in dE space it makes no sense to use the default 10 bits or so; this would only cause the minimization to thrash about needlessly.

[redacted]

[redacted]

Now do a constrained (by CMY) black matching, and interpolate the resulting four tables to give a set of transfer functions. Fix CMY at 33% each (remapped through the CMY transfer functions), to approximate a typical GCR color to black ratio. The minimization seems to be quite sensitive to the level of CMY chosen; 33% seems to work well, although in principle any fixed or even variable level could be used. Mind the total ink limits also; don't do this with ink levels going up to 400%, because they will most likely never be printed.

[redacted]



- Graphics -

Write the resulting tables out in Mathematica, ABS ScreenMaker and/or dot histogram format

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

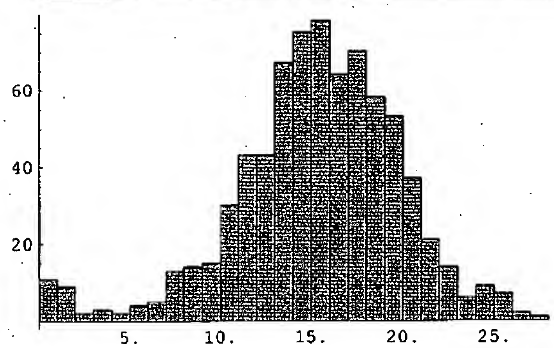
[redacted]

[redacted]

[redacted]

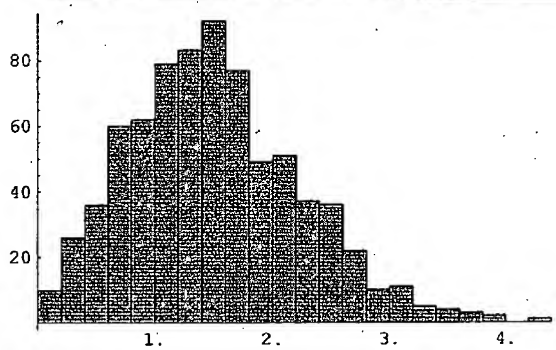
Caculate color differences over the IT8 target before and after linearization (color correction), but without reprinting and remeasuring yet, and using the models rather than the measurement data.

[redacted]



{-Graphics-, 15.2487, 27.1001, 15.5054, 4.71004}

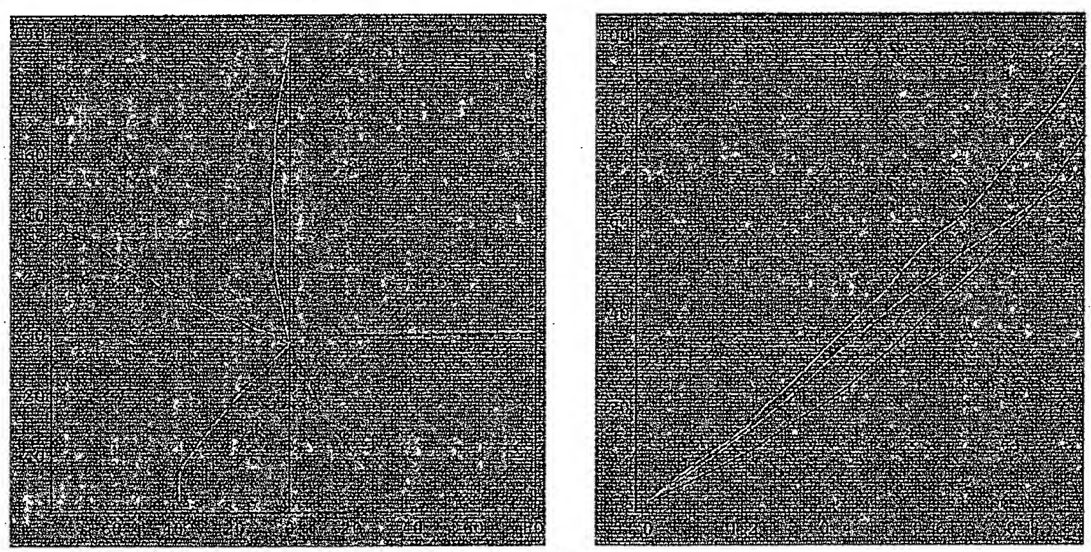
[redacted]




```
{- Graphics -, 1.50621, 4.2428, 1.44782, 0.731442}
```

Plot primary and secondary ramps as before, but after applying the linearization functions.

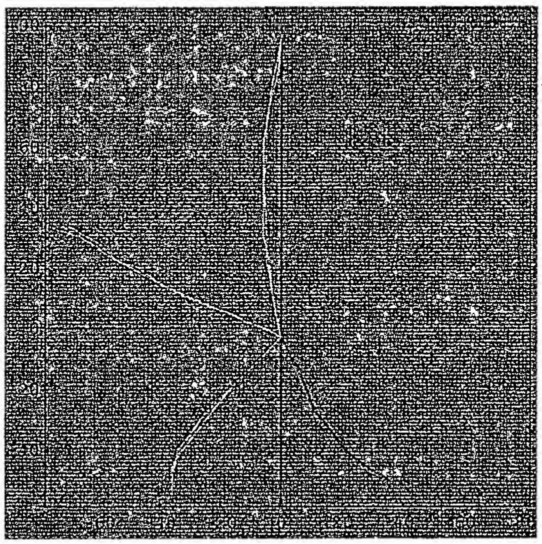
[redacted]

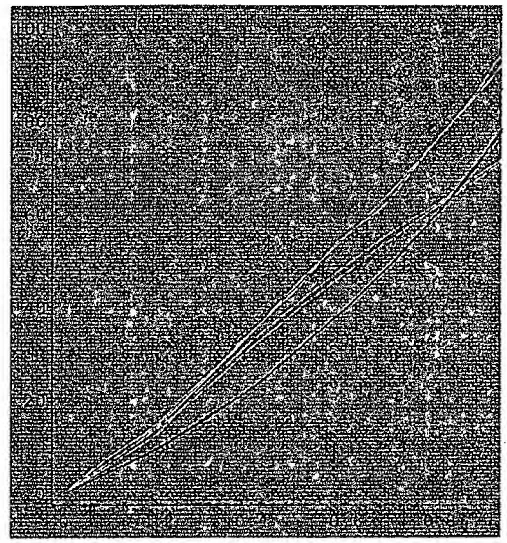


```
- GraphicsArray -
```

Show reference and result together. Note that there may be small difference induced in the indidvidual ramps in order to minimize the differences in the gray balance. Simple 1D linearization of the primaries alone would probably make the primary ramps match better, but potentially at the cost of gray balance and overall color matching.

[redacted]





- GraphicsArray -

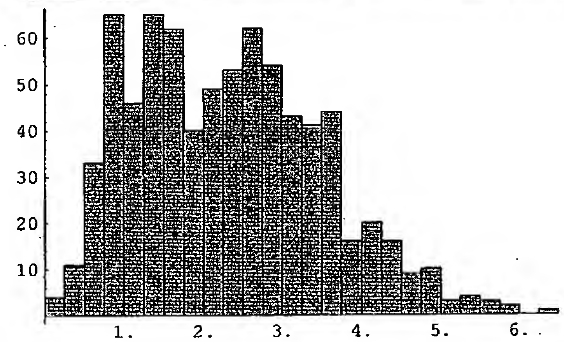
At this point we could write out a new set of "measurement" data to create a new profile corresponding to the linearized/color corrected HT. It doesn't have to be in IT8 form; since we can just turn the model crank we can generate an arbitrary number of samples, e.g. 11^4 (which would be rather time consuming to actually measure). That is, if you have faith in the model ;-)

The tranfer functions could be put into the new profile as output tables, but it's better to build them right into the (matrix) HT to avoid losing gray levels, especially if a higher bit depth (10 or 12b) version is avaiable to convert down to 8 bits while massaging the histogram to fit the required linearization. For non-matrix HTs this is more tricky.

Finally, after applying the linearizations to the new HT and reprinting the IT8 target with it, read in the measured Lab values and compare to the reference ones. This is the ultimate test for how well the model works, of course. If all is well, the results should be comparable to the ones above, which were obtained from the models only (without measurements).

[redacted]

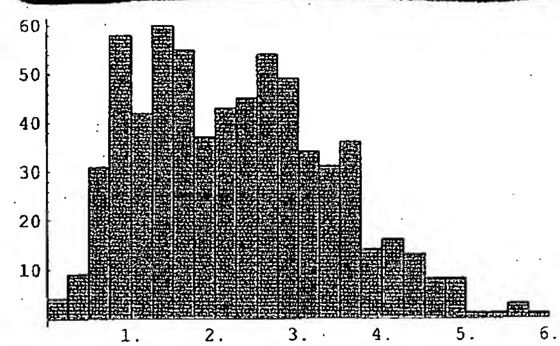
[redacted]



{0.0331662, 2.31874, 6.36356, 2.26201, 1.17569}

Select the parts of the IT8 target that are below the global ink limit for the device/ink/media under consideration, to get a more realistic picture of the color differences one might see in practice.

[redacted]



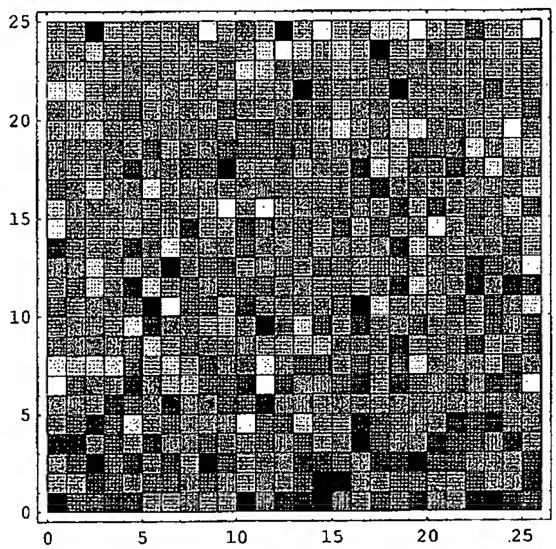
{0.0331662, 2.25058, 5.95154, 2.18417, 1.13724}

Sort the samples by descending dE to get an idea of the worst case colors

[redacted]

The following plot shows the samples with the least color differences at the top, the worst at the bottom, each colored with the corresponding CMYK color.

[redacted]



- DensityGraphics -

2. Using 1D transfer functions calculated from 1D ramps

Now compare what the result would be using traditional 1D linearizations, using the model data only (no measurements).

define the input dot percentages for a 9x21 characterization target

[redacted]

generate CIELAB data for reference and new HT, using their respective models, and convert to dE from paper

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

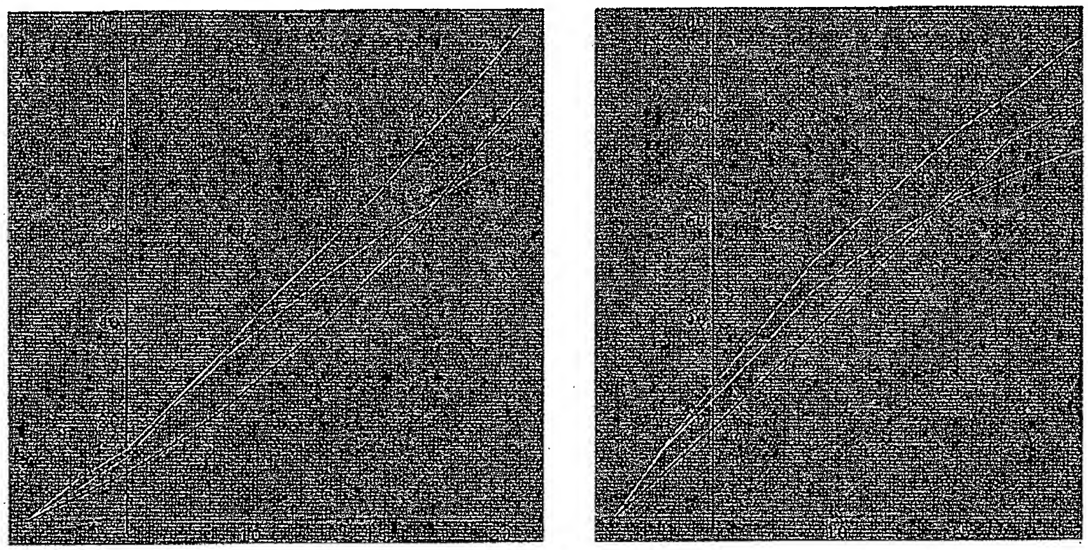
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



- GraphicsArray -

Convert to {x,y} format data using the ramp values

[redacted]

Now calculate linearization functions, using a quick and dirty numerical function inverse of interpolated functions. Varying the interpolation order will determine the smoothness of the resulting transfer functions.

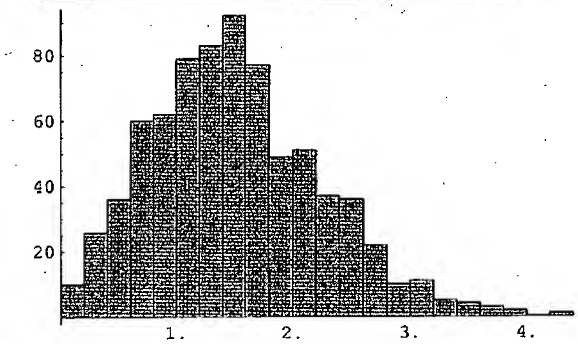
[redacted]

Caculate color differences over the IT8 target before and after linearization (color correction), but without reprinting and remeasuring yet, and using the models rather than the measurement data.

[redacted]

Previous result with gray balancing:

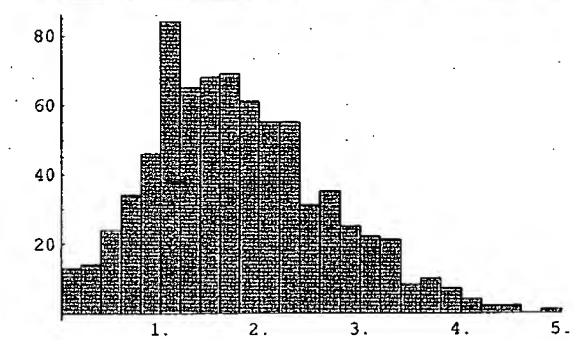
[redacted]



{- Graphics -, 1.50621, 4.2428, 1.44782, 0.731442}

New result with 1D linearization:

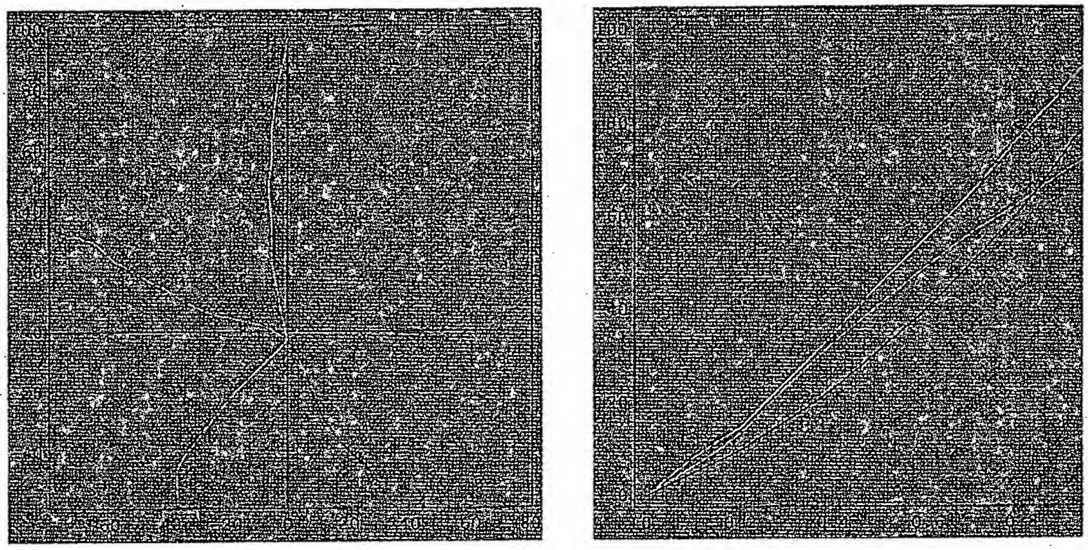
[redacted]



{- Graphics -, 1.78747, 4.97161, 1.69613, 0.874912}

Plot primary and secondary ramps as before, but after applying the new linearization functions.

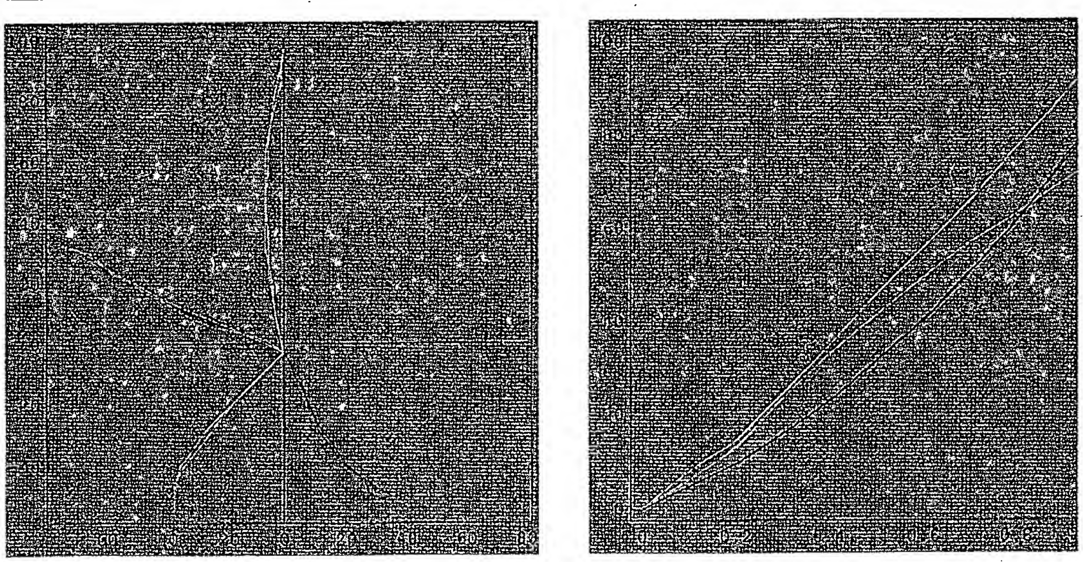
[redacted]



- GraphicsArray -

Show reference and result together. The primary ramps should match better than before.

[redacted]



- GraphicsArray -

Default HT Linearization

Johan M. Lammens

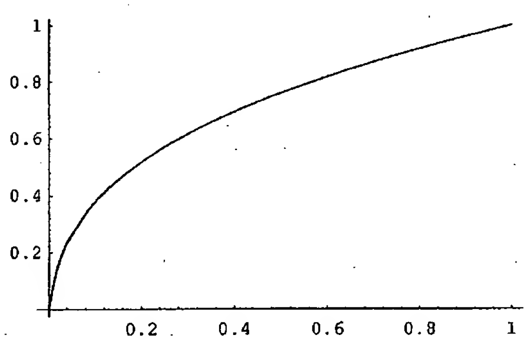
Rough default linearization for scattered dot halftones like ED or BN, based on equation for deriving L* from Y/Yn, i.e. normalized weighted reflectance, with extensions for taking dot gain into account.

[redacted]

L* from Y/Yn = r, normalized to [0,1]

[redacted]

$$\frac{-16 + 116\, r^{1/3}}{100}$$

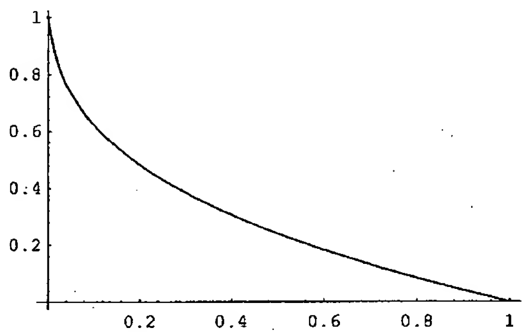


-Graphics-

Same, but in terms of digital counts (% ink)

[redacted]

$$1 + \frac{16 - 116\ d^{1/3}}{100}$$




```
{1, 0.239307, 0}
```

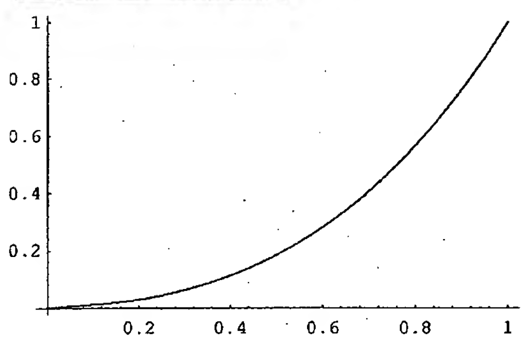
Calculate inverse of normalized L* function, taking boundary condition into account

[redacted]

[redacted]

[redacted]

[redacted]

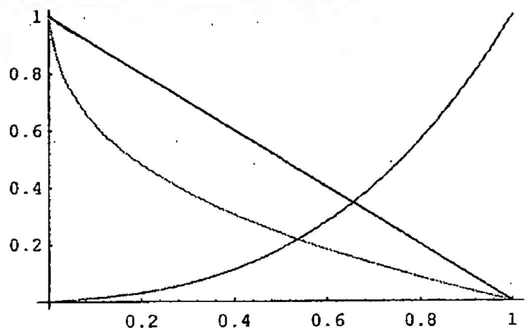



```
-Graphics-

{0, 0.184187, 1}
```

Plot L* from % ink, its inverse, and the composition of both

[redacted]




```
{1, 0.5, 0}
```

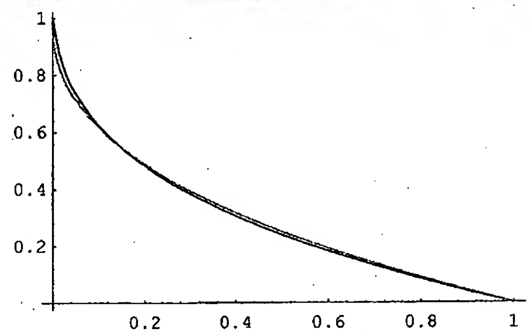
Lni is the function to use for dot growth to achieve approximately linear L*, not taking dot gain into account. An additional dot gain compensation can be applied as well.

Fit a gamma function to the L* function, using RMS error over a non-uniform sampling (the fit is worse and probably more important in the highlights).

[redacted]

[redacted]

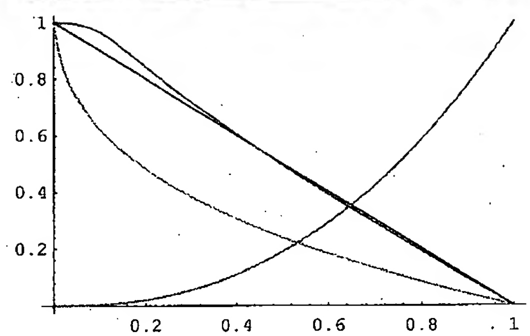
[redacted]




```
{{1, 1}, {0.621576, 0.615114}, {0.481628, 0.48695},
  {0.383458, 0.393014}, {0.305305, 0.316109}, {0.239307, 0.249808},
  {0.181618, 0.190892}, {0.130031, 0.137484}, {0.0831514, 0.0883786},
  {0.0400323, 0.042749}, {0., 0.}}
```

Try the inverse gamma function as an inverse of the L* from % ink function:

[redacted]



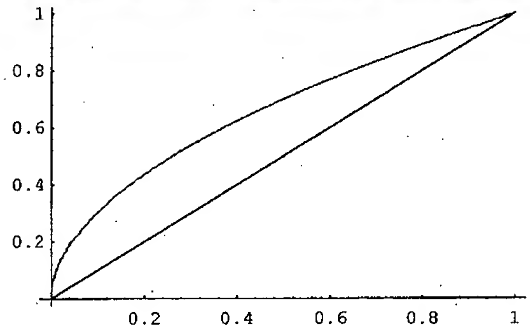

```
{1, 0.964984, 0.841883, 0.719304, 0.604644, 0.495533, 0.390651,
  0.289158, 0.190481, 0.0942004, 0.}
```

The gamma inverse is not as good, especially in the higlights, but it's a simpler function to use for the calculations below.

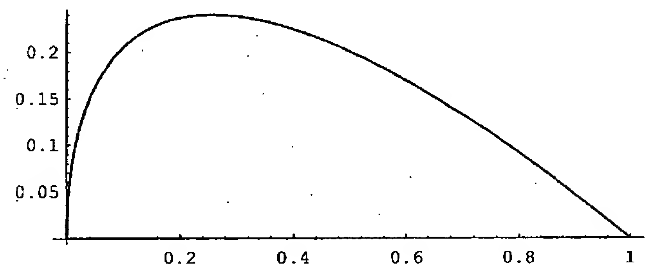
Now let's look at the effect of dot gain. Dot gain is typically specified as percent apparent dot area increase at a 50% tint; try a gamma function to extrapolate to complete range.

[redacted]

[redacted]



-Graphics-

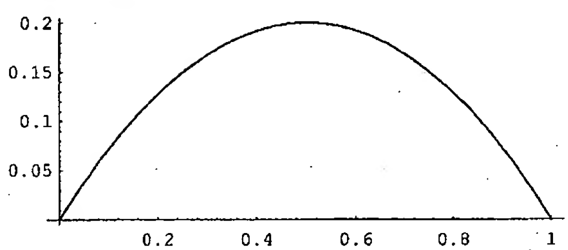


-Graphics-

This does not look like a good dot gain function; see e.g. ANSI-CGATS.6-1995 report. A quadratic function fit seems to work much better:

[redacted]

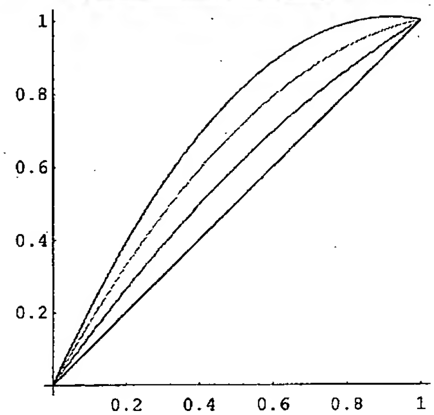
$$0.2 - 0.8\,(-0.5 + x)^2$$



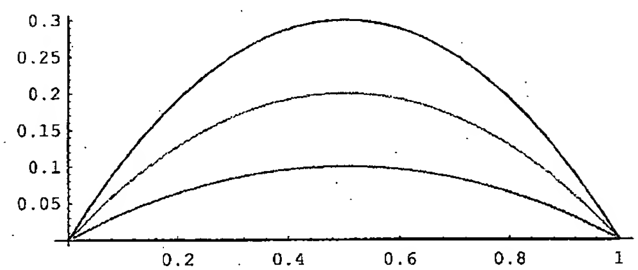
-Graphics-

Function to generate dot gain corrected % ink curves, i.e. linear plus dot gain.

[redacted]



-Graphics-



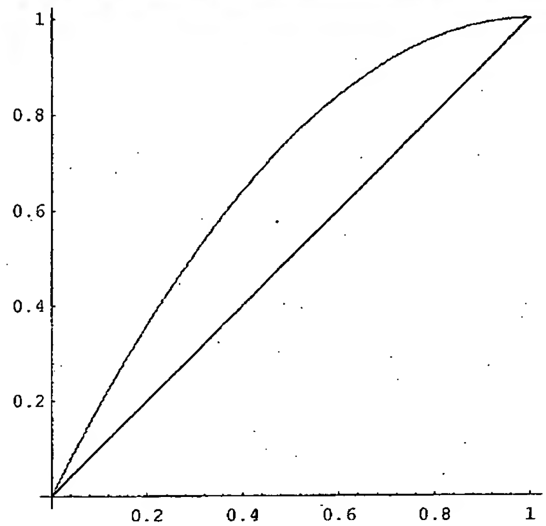
-Graphics-

The hump is on the opposite side of the one obtained with the gamma function above; this seems to be a better model for actual dot gain behavior, similar to the one PhotoShop uses when specifying only the 50% point on the curve. But the 30% curve is non-monotonic, which is not admissible - find out where the non-monotonicity sets in:

[redacted]

So the DG function should not be used with dot gain values higher than 25% (newsprint is 30%):

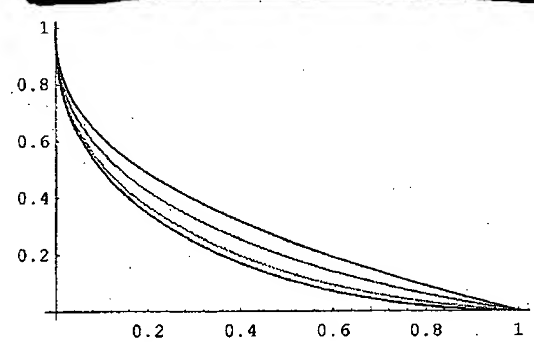
[redacted]



-Graphics-

Dot growth function for output linear in L* based on simple dot gain model and gamma function;
first, L* as a function of digital count and dot gain. Typical press dot gains range from 8% for quality coated papers to 30% for newsprint; SWOP coated is 20%.

[redacted]



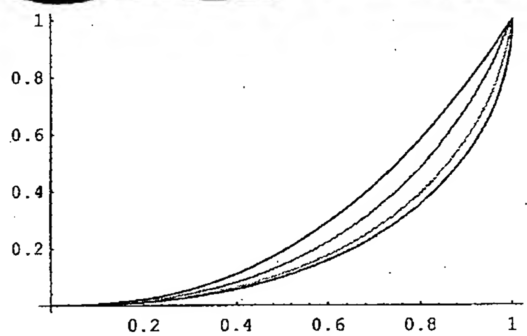
-Graphics-

Invert the L* function:

[redacted]

The inverse can obviously not be used with a dot gain value of 0, but the plain Lngi inverse can be used in stead.

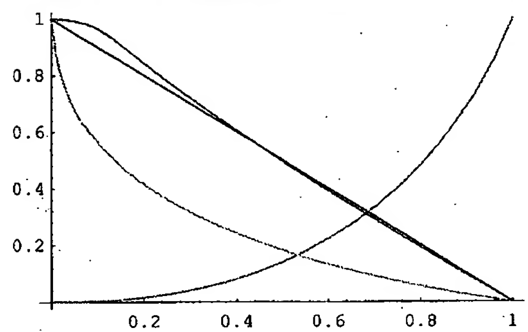
[redacted]

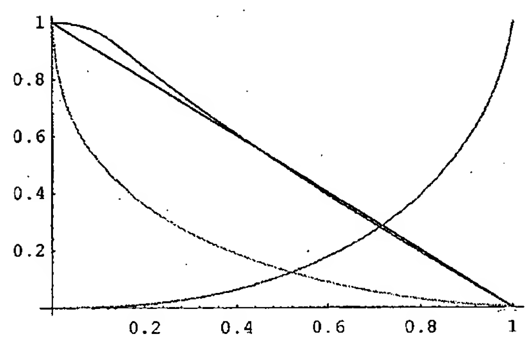


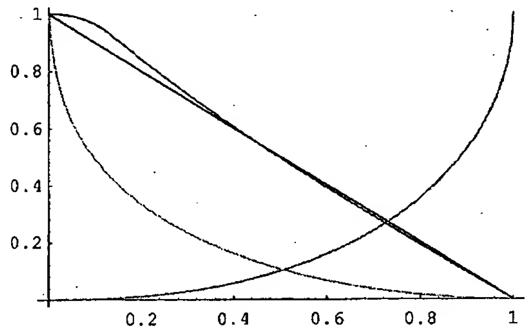
-Graphics-

Check that the inverse composed with the forward function results in linear behavior (using the original L* function):

[redacted]



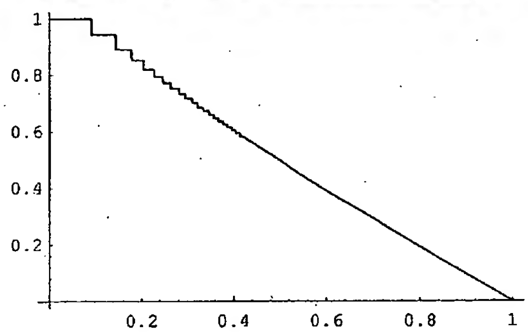


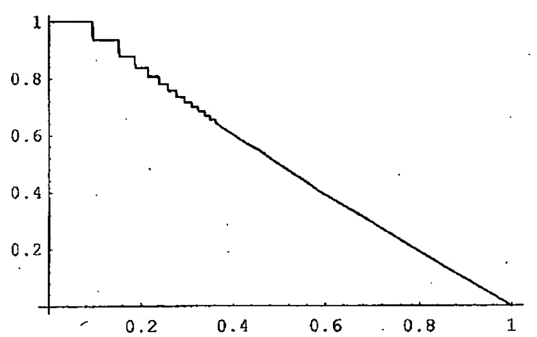


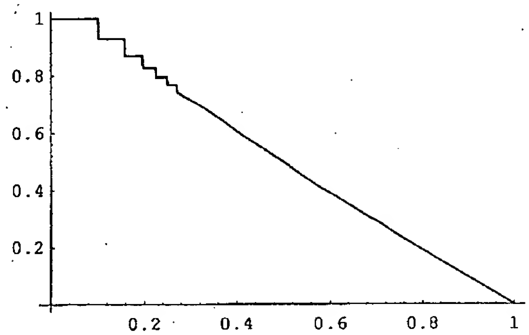
{-Graphics-, -Graphics-, -Graphics-}

To have no visible difference between digital counts differing by 1 (i.e. contouring), the difference in L* should be less than 1, the theoretical JND, or 0.01 when scaled to [0,1].
Plot normalized L* as a function of discretized 8 bit input, using the original L* function:

[redacted]








```
{-Graphics-, -Graphics-, -Graphics-}
```

Contouring is clearly visible as a stair step pattern in the highlight region. Calculate the biggest step in L* between digital counts differing by 1, which should be the first step (in the highlights). As an example, try 20% dot gain, and calculate the corresponding dE (dL*) value.

[redacted]

Function to determine the max dE between consecutive levels, given a certain number of bits of precision

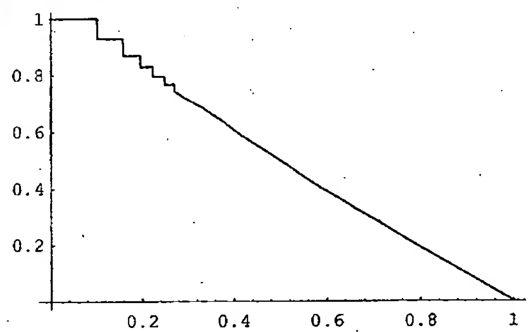
[redacted]

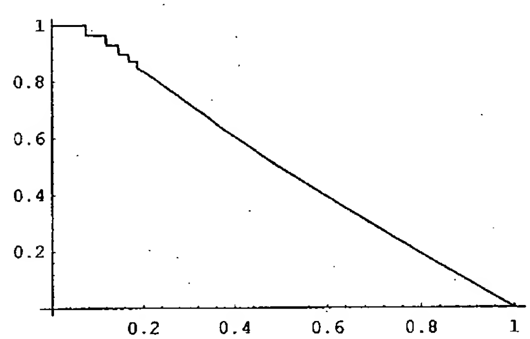
Find the number of bits required for a given maximum dE at a given level of dot gain, and plot linearity as a function of dot gain and bit depth.

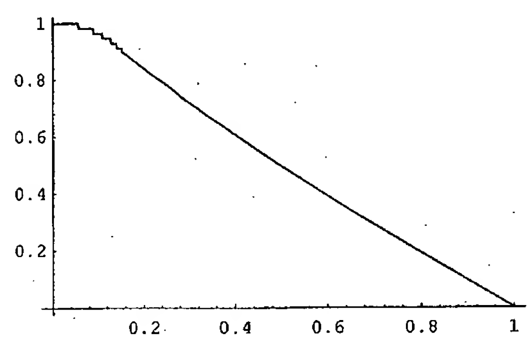
[redacted]

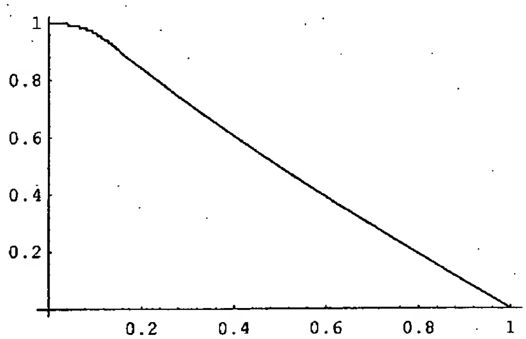
So finally it seems that in order to be able to visually linearize a halftone that is linear in the number of dots (like standard error diffusion) without risking any contoutring and taking a reasonable dot gain into account, we need at least 11 bits in the transfer function and in the halftone. The plots below visualize the solution found.

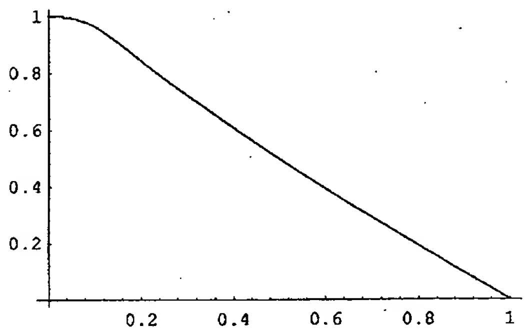
[redacted]












```
{-Graphics-, -Graphics-, -Graphics-, -Graphics-, -Graphics-}
```

Conclusion

To be safe, we need at least 11 bits of grayscale resolution when linearizing arbitrary halftones on reasonable media. To have some design margin and because it's easier to deal with in the digital domain, I'd recommend 12 bits.

JML

Read a HT matrix and generate its dot (threshold) histogram (and ABS dot area file).

Johan Lammens [redacted]

[redacted]

decoding functions for ASCIIHex format

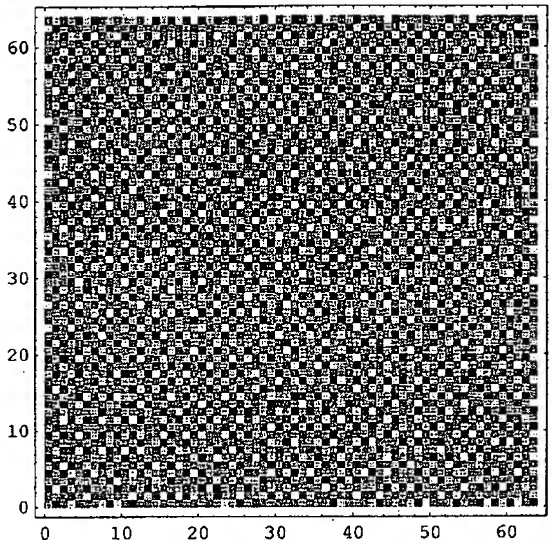
[redacted]

Read input HT, raw or asciihex format, possibly invert from additive to subtractive interpretation

[redacted]

Display in 2D matrix form

[redacted]



Write out in Alp monochrome form and for FFT analysis

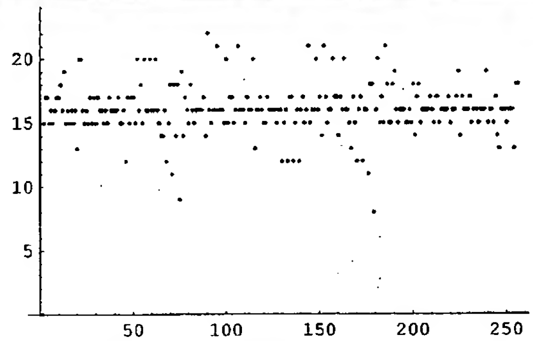
[redacted]

Write out in raw form

[redacted]

Calculate histogram and write to ascii file, mma format

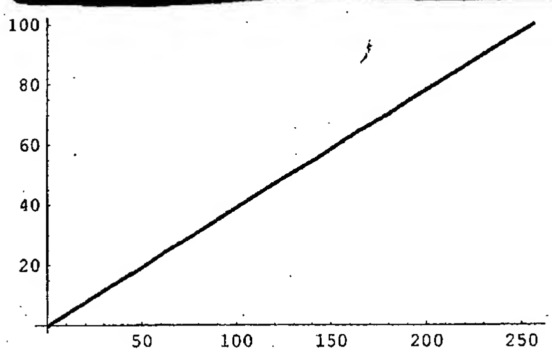
[redacted]



[redacted]

Convert to "dot area" (assuming perfect square dots) and write to file

[redacted]




```
{256, 0., 100., 12827.7}
```

[redacted]